University at Buffalo
The State University of New York

Executive Summary: Creating a Simple ETL Pipeline

By

POOJA DAYANAND KABADI - 50575012

Table of Contents

List of Figures

1. Overview

The main activity is to implement a basic ETL(Extract, Transform, Load) pipeline to handle data integration for a sample database. Using Apache Hop, Oracle Cloud, and SQL queries, I developed processes to load dimension and fact tables. This report summarizes the design choices, database schema, and major steps involved in creating the pipeline, along with insights into potential improvements

2. ETL Pipeline Design and Database Diagram

The ETL pipeline was created in Apache Hop to enable the seamless movement of data from multiple sources into a consolidated database. A diagram of the database schema was generated, showing key entities such as **Dim_Date, Dim_Product, Dim_Customer, and the main fact table.** This schema aimed to support typical analytical queries, with each dimension table containing relevant details about dates, products, and customers.

3. Process Steps and Key Components

- **Loading Dimension Tables:**
  - **Dim_Date:** Rows were loaded by executing SQL queries to retrieve relevant date information.
  - **Dim_Product:** Rows were inserted using DDL, and screenshots of successful SQL executions are attached.
  - **Dim_Customer:** Rows were inserted using DDL, with attached screenshots of successful SQL executions, and output was verified to ensure accurate data capture.
- **Pipeline Execution and Validation:**
  - **Apache Hop Pipeline:** Configured using necessary transformations, each validated through visual monitoring in Apache Hop. A screenshot of the pipeline setup is attached to show the flow and data processing steps.
  - **Oracle Cloud Validation:** SQL queries were run to validate the data in Oracle Cloud, with screenshots confirming the correct loading of data.
- **Fact Table Integration:**

  For the fact table, a stream connector was used to link data from multiple sources effectively. Screenshots of the entire flow, including the stream connector and its configuration, are included. This approach allowed seamless and efficient data integration, ensuring the fact table received up-to-date, accurate data through continuous data streams.

4. Challenges

I encountered several challenges related to modifying database parallel to the ETL pipeline. Initially, I faced an issue where the pipeline encountered errors due to table modifications in the database. To address this, I created a new database from scratch, ensuring a stable structure before implementing the pipeline.

Later in the process, after making some minor adjustments to the database schema, the pipeline again encountered issues. This time, the problem was resolved by committing the changes in the database before re-running the pipeline. This experience underscored the importance of committing schema updates consistently to avoid disruptions during ETL execution.

5. Missing Components

This database model is missing foreign keys and referential constraints between the fact and dimension tables, which are typically expected in a star schema design. These constraints ensure data integrity by linking dimension tables to the fact table through key relationships. They might be missing here to simplify the model, either to speed up data loading or because of a decision to handle integrity through ETL processes rather than within the database schema itself. Also, the database schema currently supports essential analytical functions. However, an additional dimension table, such as Dim_Location for geographical data, might enhance the system's analytical depth. These components may have been excluded due to initial scope of requirements but could add value in future expansions.

6. Conclusion

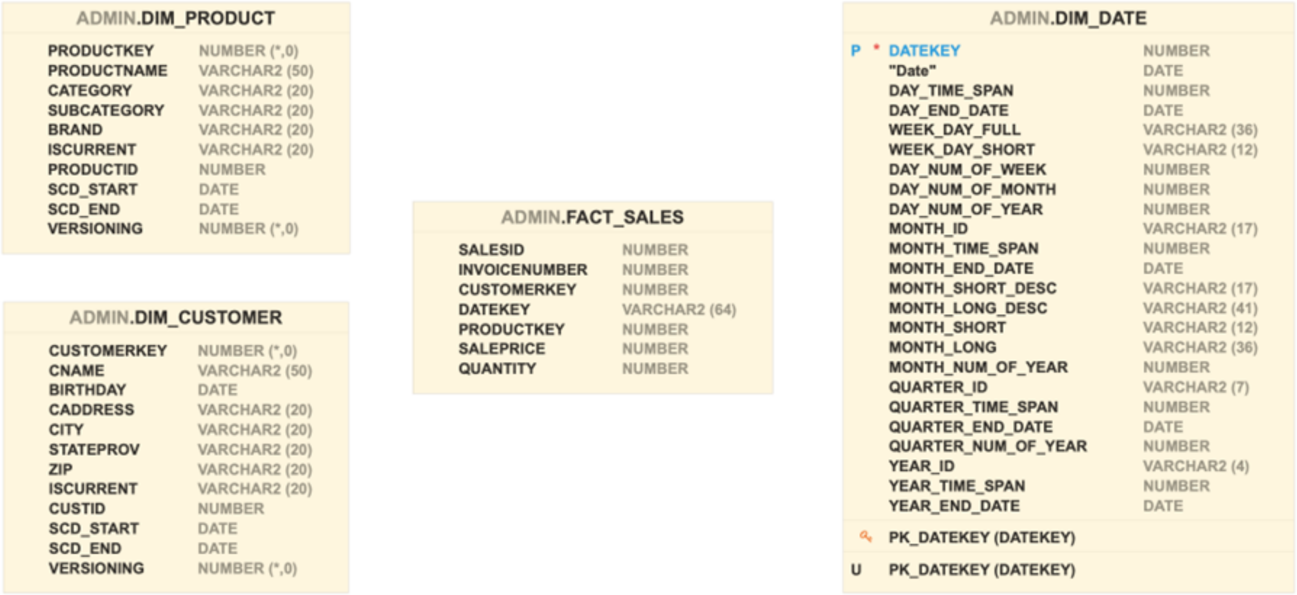
The ETL pipeline implemented in this project successfully facilitated data integration from multiple sources into a consolidated database, demonstrating the efficiency and reliability of Apache Hop and Oracle Cloud for building and maintaining data workflows. By following structured steps to load dimension and fact tables, this pipeline supported essential analytical functions, with each design choice tailored to meet the initial scope.

While the pipeline effectively handled the basic needs of the sample database, several areas for improvement have been identified. Adding foreign keys and referential constraints would strengthen data integrity, linking fact and dimension tables and ensuring consistency across the schema. Additionally, the introduction of a Dim_Location table could provide valuable insights into geographical trends, enhancing the model's analytical potential.

This project highlighted the importance of managing schema updates, committing changes carefully to avoid disruptions. It underscored how a flexible, well-structured ETL pipeline can be expanded to support more advanced analytics, with future improvements likely to enhance the system’s robustness and analytical capabilities.

7. Database Diagram

Generate a diagram of the database



ADMIN.DIM_PRODUCT

PRODUCTKEY	NUMBER (*,0)
PRODUCTNAME	VARCHAR2 (50)
CATEGORY	VARCHAR2 (20)
SUBCATEGORY	VARCHAR2 (20)
BRAND	VARCHAR2 (20)
ISCURRENT	VARCHAR2 (20)
PRODUCTID	NUMBER
SCD_START	DATE
SCD_END	DATE
VERSIONING	NUMBER (*,0)

ADMIN.DIM_CUSTOMER

CUSTOMERKEY	NUMBER (*,0)
CNAME	VARCHAR2 (50)
BIRTHDAY	DATE
CADDRESS	VARCHAR2 (20)
CITY	VARCHAR2 (20)
STATEPROV	VARCHAR2 (20)
ZIP	VARCHAR2 (20)
ISCURRENT	VARCHAR2 (20)
CUSTID	NUMBER
SCD_START	DATE
SCD_END	DATE
VERSIONING	NUMBER (*,0)

ADMIN.FACT_SALES

SALESID	NUMBER
INVOICENUMBER	NUMBER
CUSTOMERKEY	NUMBER
DATEKEY	VARCHAR2 (64)
PRODUCTKEY	NUMBER
SALEPRICE	NUMBER
QUANTITY	NUMBER

ADMIN.DIM_DATE

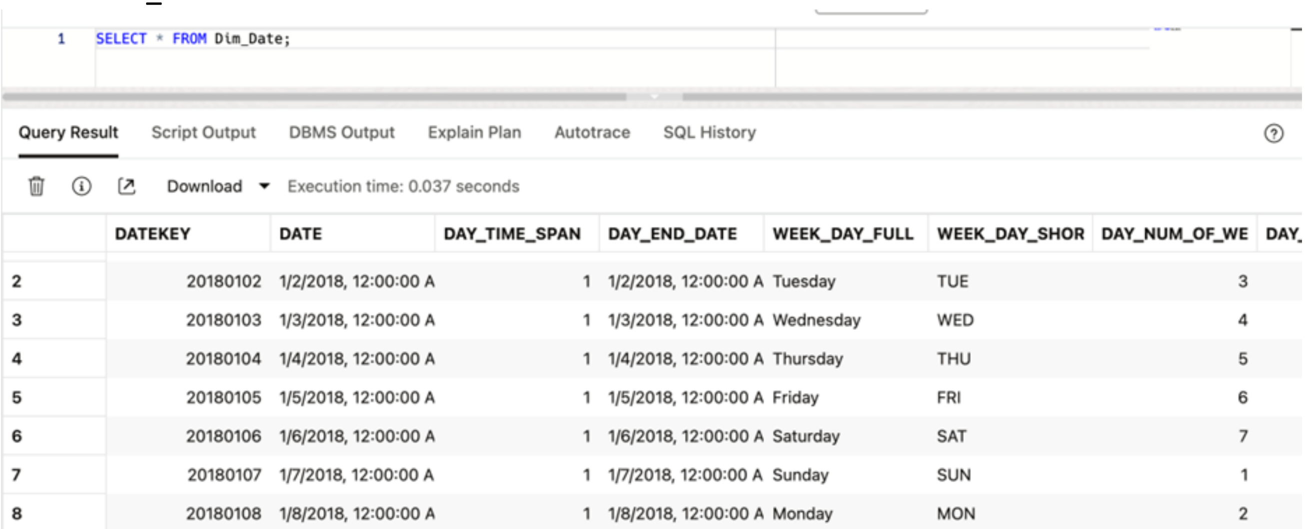
P *	DATEKEY	NUMBER
	"Date"	DATE
	DAY_TIME_SPAN	NUMBER
	DAY_END_DATE	DATE
	WEEK_DAY_FULL	VARCHAR2 (36)
	WEEK_DAY_SHORT	VARCHAR2 (12)
	DAY_NUM_OF_WEEK	NUMBER
	DAY_NUM_OF_MONTH	NUMBER
	DAY_NUM_OF_YEAR	NUMBER
	MONTH_ID	VARCHAR2 (17)
	MONTH_TIME_SPAN	NUMBER
	MONTH_END_DATE	DATE
	MONTH_SHORT_DESC	VARCHAR2 (17)
	MONTH_LONG_DESC	VARCHAR2 (41)
	MONTH_SHORT	VARCHAR2 (12)
	MONTH_LONG	VARCHAR2 (36)
	MONTH_NUM_OF_YEAR	NUMBER
	QUARTER_ID	VARCHAR2 (7)
	QUARTER_TIME_SPAN	NUMBER
	QUARTER_END_DATE	DATE
	QUARTER_NUM_OF_YEAR	NUMBER
	YEAR_ID	VARCHAR2 (4)
	YEAR_TIME_SPAN	NUMBER
	YEAR_END_DATE	DATE
	PK_DATEKEY (DATEKEY)	
U	PK_DATEKEY (DATEKEY)	

Figure 1 Database Diagram

- **Based on the diagram generated, what is this database missing that you'd expect to see? Why might it be missing this component?**

  This database model lacks foreign keys and referential constraints between the fact and dimension tables, which are standard elements in a typical star schema design. Such constraints are essential for maintaining data integrity by linking dimension tables (e.g., DIM_CUSTOMER, DIM_PRODUCT, DIM_DATE) to the fact table (FACT_SALES) through defined key relationships. Their absence here may be intentional, possibly to simplify the model, speed up data loading, or shift the responsibility for data integrity to the ETL processes rather than enforcing it within the database schema itself.

8. Dim_Date



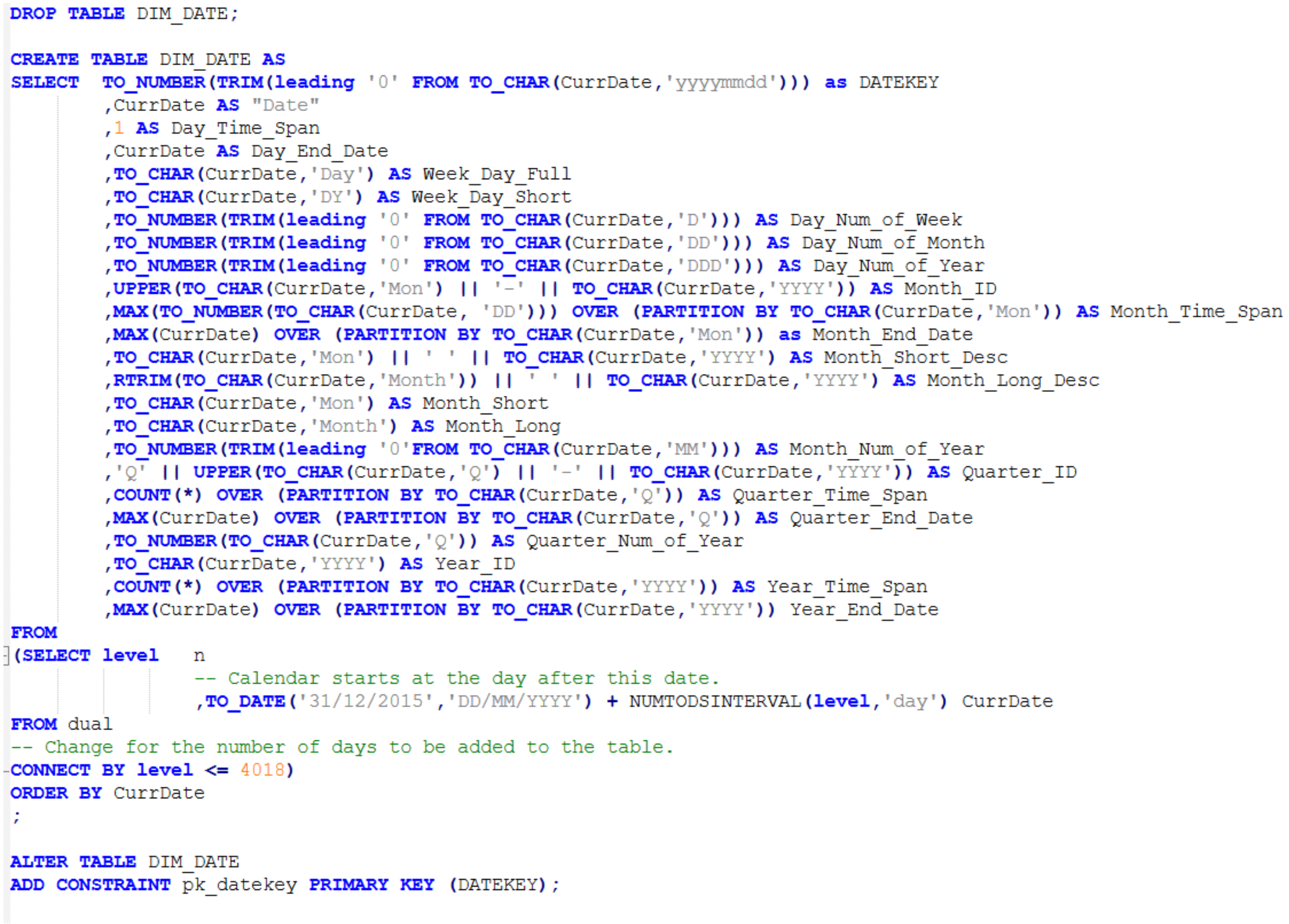
```
SELECT * FROM Dim_Date;
```

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DAY_NUM_OF_WE	DAY_
2	20180102	1/2/2018, 12:00:00 A	1	1/2/2018, 12:00:00 A	Tuesday	TUE	3	
3	20180103	1/3/2018, 12:00:00 A	1	1/3/2018, 12:00:00 A	Wednesday	WED	4	
4	20180104	1/4/2018, 12:00:00 A	1	1/4/2018, 12:00:00 A	Thursday	THU	5	
5	20180105	1/5/2018, 12:00:00 A	1	1/5/2018, 12:00:00 A	Friday	FRI	6	
6	20180106	1/6/2018, 12:00:00 A	1	1/6/2018, 12:00:00 A	Saturday	SAT	7	
7	20180107	1/7/2018, 12:00:00 A	1	1/7/2018, 12:00:00 A	Sunday	SUN	1	
8	20180108	1/8/2018, 12:00:00 A	1	1/8/2018, 12:00:00 A	Monday	MON	2	

Figure 2 Dim_Date Table

9. Dim_Date Updated

The client isn’t happy with the values in the date dimension table and has asked to redo the dimension table in the data warehouse to begin in January, 2016 and end in December, 2026, updated script is as follows

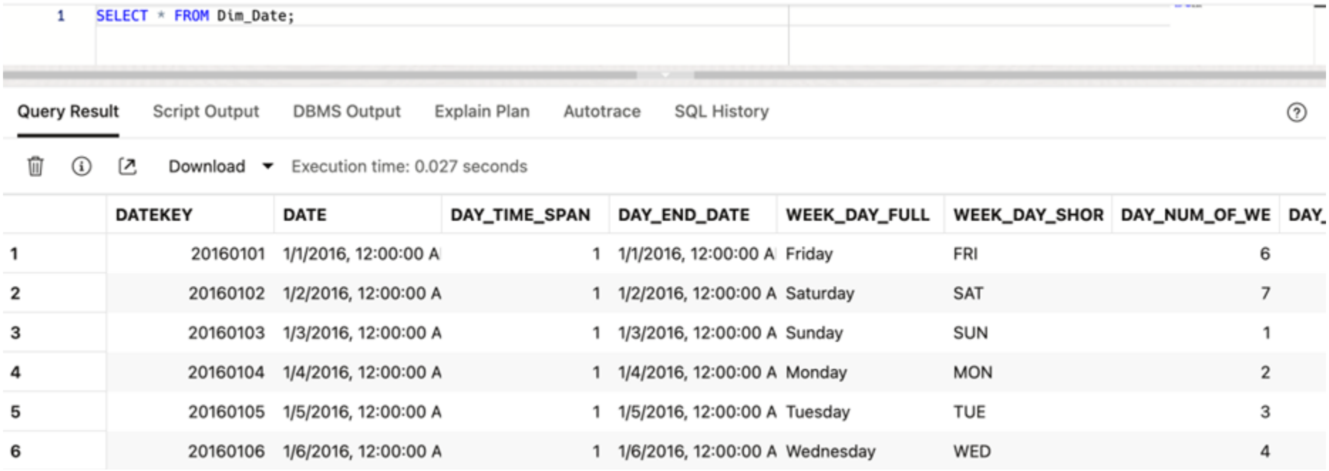


```
DROP TABLE DIM_DATE;

CREATE TABLE DIM_DATE AS
SELECT  TO_NUMBER(TRIM(leading '0' FROM TO_CHAR(CurrDate,'yyyymmdd'))) as DATEKEY
        ,CurrDate AS "Date"
        ,1 AS Day_Time_Span
        ,CurrDate AS Day_End_Date
        ,TO_CHAR(CurrDate,'Day') AS Week_Day_Full
        ,TO_CHAR(CurrDate,'DY') AS Week_Day_Short
        ,TO_NUMBER(TRIM(leading '0' FROM TO_CHAR(CurrDate,'D'))) AS Day_Num_of_Week
        ,TO_NUMBER(TRIM(leading '0' FROM TO_CHAR(CurrDate,'DD'))) AS Day_Num_of_Month
        ,TO_NUMBER(TRIM(leading '0' FROM TO_CHAR(CurrDate,'DDD'))) AS Day_Num_of_Year
        ,UPPER(TO_CHAR(CurrDate,'Mon') || '-' || TO_CHAR(CurrDate,'YYYY')) AS Month_ID
        ,MAX(TO_NUMBER(TO_CHAR(CurrDate, 'DD'))) OVER (PARTITION BY TO_CHAR(CurrDate,'Mon')) AS Month_Time_Span
        ,MAX(CurrDate) OVER (PARTITION BY TO_CHAR(CurrDate,'Mon')) as Month_End_Date
        ,TO_CHAR(CurrDate,'Mon') || ' ' || TO_CHAR(CurrDate,'YYYY') AS Month_Short_Desc
        ,RTRIM(TO_CHAR(CurrDate,'Month')) || ' ' || TO_CHAR(CurrDate,'YYYY') AS Month_Long_Desc
        ,TO_CHAR(CurrDate,'Mon') AS Month_Short
        ,TO_CHAR(CurrDate,'Month') AS Month_Long
        ,TO_NUMBER(TRIM(leading '0'FROM TO_CHAR(CurrDate,'MM'))) AS Month_Num_of_Year
        ,'Q' || UPPER(TO_CHAR(CurrDate,'Q') || '-' || TO_CHAR(CurrDate,'YYYY')) AS Quarter_ID
        ,COUNT(*) OVER (PARTITION BY TO_CHAR(CurrDate,'Q')) AS Quarter_Time_Span
        ,MAX(CurrDate) OVER (PARTITION BY TO_CHAR(CurrDate,'Q')) AS Quarter_End_Date
        ,TO_NUMBER(TO_CHAR(CurrDate,'Q')) AS Quarter_Num_of_Year
        ,TO_CHAR(CurrDate,'YYYY') AS Year_ID
        ,COUNT(*) OVER (PARTITION BY TO_CHAR(CurrDate,'YYYY')) AS Year_Time_Span
        ,MAX(CurrDate) OVER (PARTITION BY TO_CHAR(CurrDate,'YYYY')) Year_End_Date
FROM
(SELECT level   n
                -- Calendar starts at the day after this date.
                ,TO_DATE('31/12/2015','DD/MM/YYYY') + NUMTODSINTERVAL(level,'day') CurrDate
FROM dual
-- Change for the number of days to be added to the table.
CONNECT BY level <= 4018)
ORDER BY CurrDate
;

ALTER TABLE DIM_DATE
ADD CONSTRAINT pk_datekey PRIMARY KEY (DATEKEY);
```

Figure 3 Dim_Date DDL Updated script

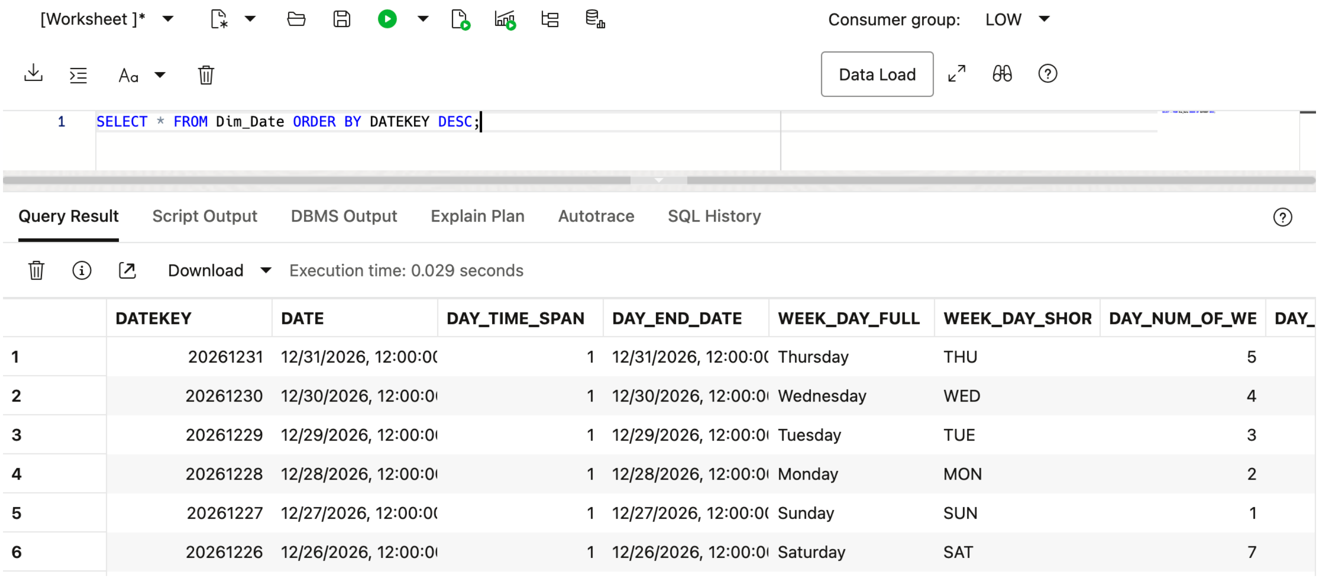


```
SELECT * FROM Dim_Date;
```

Execution time: 0.027 seconds

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DAY_NUM_OF_WE	DAY_
1	20160101	1/1/2016, 12:00:00 A	1	1/1/2016, 12:00:00 A	Friday	FRI	6	
2	20160102	1/2/2016, 12:00:00 A	1	1/2/2016, 12:00:00 A	Saturday	SAT	7	
3	20160103	1/3/2016, 12:00:00 A	1	1/3/2016, 12:00:00 A	Sunday	SUN	1	
4	20160104	1/4/2016, 12:00:00 A	1	1/4/2016, 12:00:00 A	Monday	MON	2	
5	20160105	1/5/2016, 12:00:00 A	1	1/5/2016, 12:00:00 A	Tuesday	TUE	3	
6	20160106	1/6/2016, 12:00:00 A	1	1/6/2016, 12:00:00 A	Wednesday	WED	4	

Figure 4 Dim_Date Updated(Date ascending order)



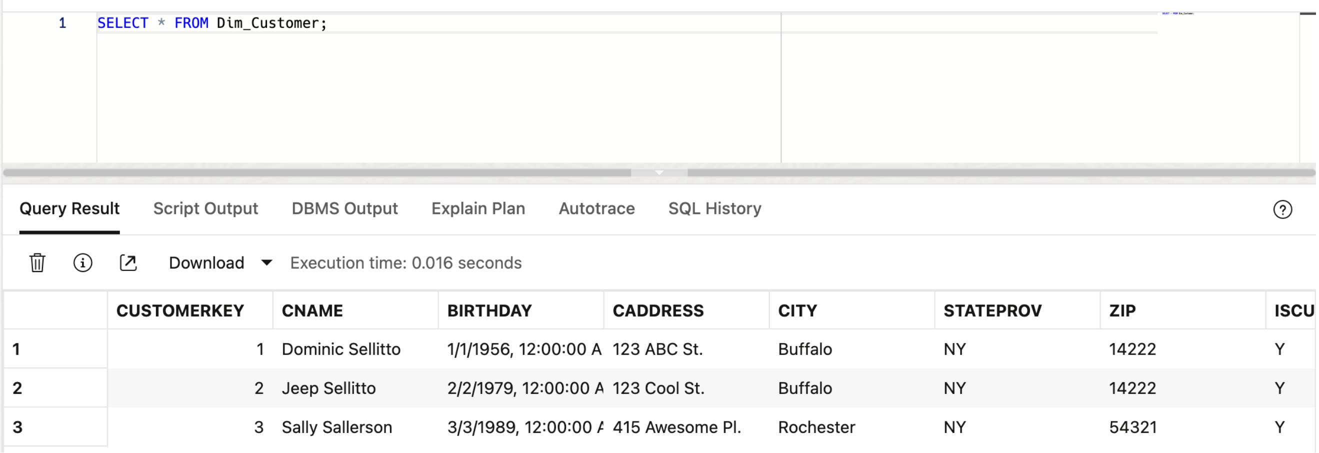
```
SELECT * FROM Dim_Date ORDER BY DATEKEY DESC;
```

Execution time: 0.029 seconds

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DAY_NUM_OF_WE	DAY_
1	20261231	12/31/2026, 12:00:0(	1	12/31/2026, 12:00:0(	Thursday	THU	5	
2	20261230	12/30/2026, 12:00:0	1	12/30/2026, 12:00:0	Wednesday	WED	4	
3	20261229	12/29/2026, 12:00:0	1	12/29/2026, 12:00:0	Tuesday	TUE	3	
4	20261228	12/28/2026, 12:00:0	1	12/28/2026, 12:00:0	Monday	MON	2	
5	20261227	12/27/2026, 12:00:0(	1	12/27/2026, 12:00:0(	Sunday	SUN	1	
6	20261226	12/26/2026, 12:00:0	1	12/26/2026, 12:00:0	Saturday	SAT	7	

Figure 5 Dim_Date Updated(Date descending order)

10. Dim_Customer



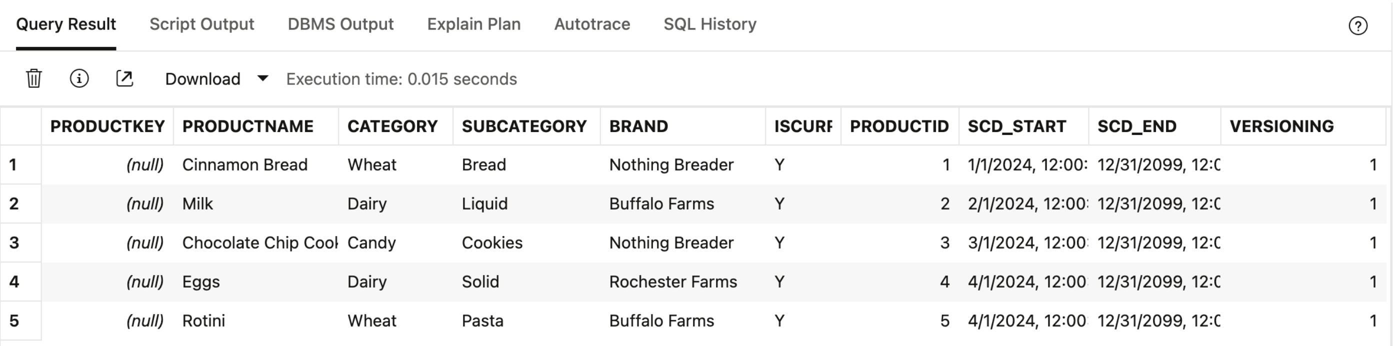
```
SELECT * FROM Dim_Customer;
```

Execution time: 0.016 seconds

	CUSTOMERKEY	CNAME	BIRTHDAY	CADDRESS	CITY	STATEPROV	ZIP	ISCU
1	1	Dominic Sellitto	1/1/1956, 12:00:00 A	123 ABC St.	Buffalo	NY	14222	Y
2	2	Jeep Sellitto	2/2/1979, 12:00:00 A	123 Cool St.	Buffalo	NY	14222	Y
3	3	Sally Sallerson	3/3/1989, 12:00:00 A	415 Awesome Pl.	Rochester	NY	54321	Y

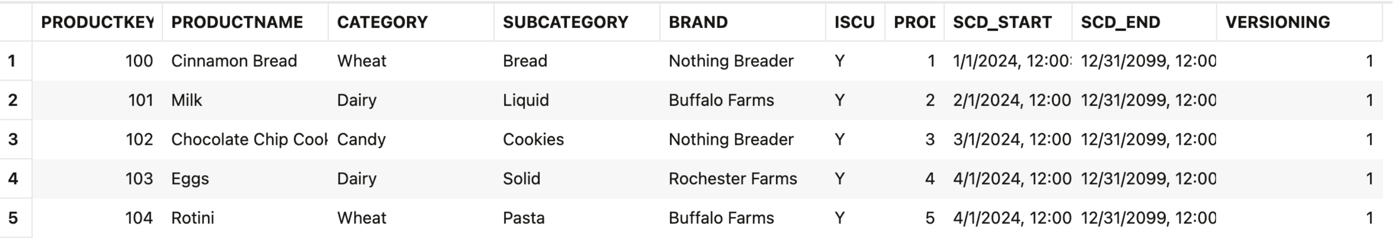
Figure 6 Dim_Customer

11. Dim_Product



	PRODUCTKEY	PRODUCTNAME	CATEGORY	SUBCATEGORY	BRAND	ISCURI	PRODUCTID	SCD_START	SCD_END	VERSIONING
1	(null)	Cinnamon Bread	Wheat	Bread	Nothing Breader	Y	1	1/1/2024, 12:00:	12/31/2099, 12:C	1
2	(null)	Milk	Dairy	Liquid	Buffalo Farms	Y	2	2/1/2024, 12:00:	12/31/2099, 12:C	1
3	(null)	Chocolate Chip Cool	Candy	Cookies	Nothing Breader	Y	3	3/1/2024, 12:00	12/31/2099, 12:C	1
4	(null)	Eggs	Dairy	Solid	Rochester Farms	Y	4	4/1/2024, 12:00	12/31/2099, 12:C	1
5	(null)	Rotini	Wheat	Pasta	Buffalo Farms	Y	5	4/1/2024, 12:00	12/31/2099, 12:C	1

Figure 7 Dim_Product(without sequence)



	PRODUCTKEY	PRODUCTNAME	CATEGORY	SUBCATEGORY	BRAND	ISCU	PROI	SCD_START	SCD_END	VERSIONING
1	100	Cinnamon Bread	Wheat	Bread	Nothing Breader	Y	1	1/1/2024, 12:00:	12/31/2099, 12:00	1
2	101	Milk	Dairy	Liquid	Buffalo Farms	Y	2	2/1/2024, 12:00	12/31/2099, 12:00	1
3	102	Chocolate Chip Cool	Candy	Cookies	Nothing Breader	Y	3	3/1/2024, 12:00	12/31/2099, 12:00	1
4	103	Eggs	Dairy	Solid	Rochester Farms	Y	4	4/1/2024, 12:00	12/31/2099, 12:00	1
5	104	Rotini	Wheat	Pasta	Buffalo Farms	Y	5	4/1/2024, 12:00	12/31/2099, 12:00	1

Figure 8 Dim_Product(with Sequencing)

12. Pipeline 1- Text file output

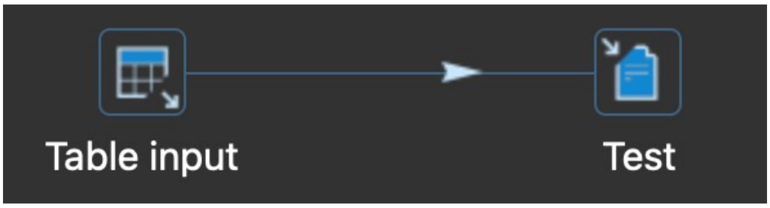
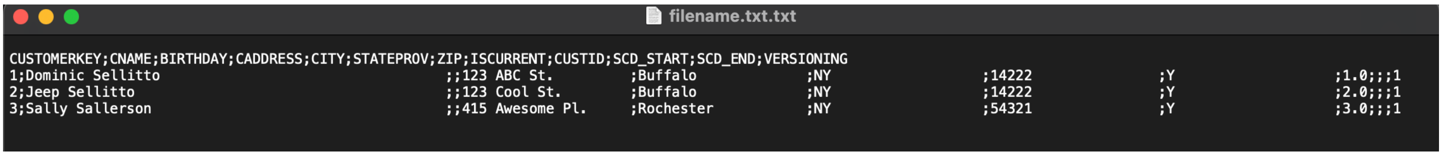


Figure 9 Pipeline 1-Text File Output



```
CUSTOMERKEY;CNAME;BIRTHDAY;CADDRESS;CITY;STATEPROV;ZIP;ISCURRENT;CUSTID;SCD_START;SCD_END;VERSIONING
1;Dominic Sellitto                       ;;123 ABC St.     ;Buffalo          ;NY              ;14222           ;Y                ;1.0;;;1
2;Jeep Sellitto                          ;;123 Cool St.    ;Buffalo          ;NY              ;14222           ;Y                ;2.0;;;1
3;Sally Sallerson                        ;;415 Awesome Pl. ;Rochester        ;NY              ;54321           ;Y                ;3.0;;;1
```

Figure 10 Pipeline 1-Text File Output

13. Pipeline 2-Product Dimension Update

This pipeline flow is designed to update the product data warehouse table based on changes in an Excel file. Here, all options are set to insert new records, except for the product name, which is configured to "punch through" updates. This means that if a dimension table already contains the same combination of values for a given product name, only the existing record will be updated with the new values, effectively revising its version.

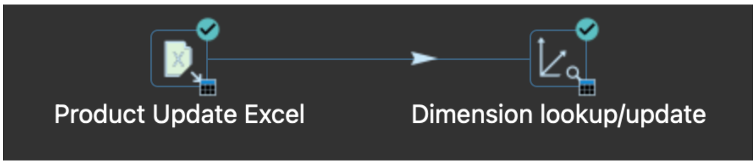
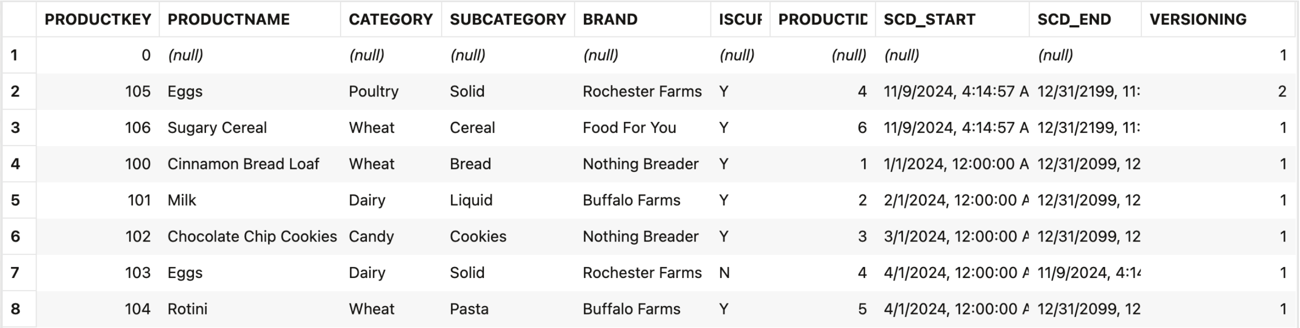


Figure 11 Pipeline 2-Product Update



	PRODUCTKEY	PRODUCTNAME	CATEGORY	SUBCATEGORY	BRAND	ISCUI	PRODUCTIE	SCD_START	SCD_END	VERSIONING
1	0	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	1
2	105	Eggs	Poultry	Solid	Rochester Farms	Y	4	11/9/2024, 4:14:57 A	12/31/2199, 11:	2
3	106	Sugary Cereal	Wheat	Cereal	Food For You	Y	6	11/9/2024, 4:14:57 A	12/31/2199, 11:	1
4	100	Cinnamon Bread Loaf	Wheat	Bread	Nothing Breader	Y	1	1/1/2024, 12:00:00 A	12/31/2099, 12	1
5	101	Milk	Dairy	Liquid	Buffalo Farms	Y	2	2/1/2024, 12:00:00 A	12/31/2099, 12	1
6	102	Chocolate Chip Cookies	Candy	Cookies	Nothing Breader	Y	3	3/1/2024, 12:00:00 A	12/31/2099, 12	1
7	103	Eggs	Dairy	Solid	Rochester Farms	N	4	4/1/2024, 12:00:00 A	11/9/2024, 4:1	1
8	104	Rotini	Wheat	Pasta	Buffalo Farms	Y	5	4/1/2024, 12:00:00 A	12/31/2099, 12	1

Figure 12 Dim_Product Updated via Apache hop flow

14. Pipeline 3-Customer Dimension Update

This pipeline flow is designed to update the customer data warehouse table based on changes in an Excel file. Here, all options are set to insert new records, except for the customer name, which is configured to "punch through" updates. This means that if a dimension table already contains the same combination of values for a given product name, only the existing record will be updated with the new values, effectively revising its version.

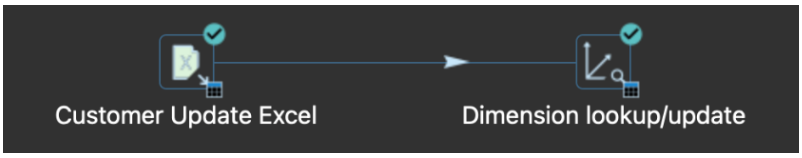
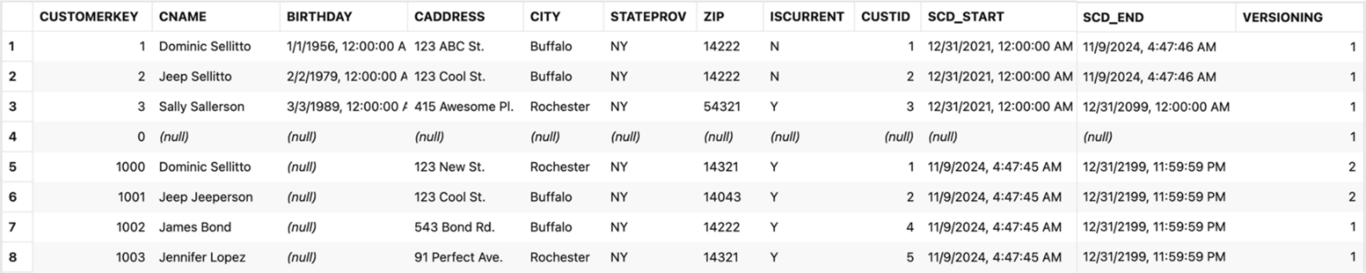


Figure 13 Pipeline 3-Customer Update



	CUSTOMERKEY	CNAME	BIRTHDAY	CADDRESS	CITY	STATEPROV	ZIP	ISCURRENT	CUSTID	SCD_START	SCD_END	VERSIONING
1	1	Dominic Sellitto	1/1/1956, 12:00:00 A	123 ABC St.	Buffalo	NY	14222	N	1	12/31/2021, 12:00:00 AM	11/9/2024, 4:47:46 AM	1
2	2	Jeep Sellitto	2/2/1979, 12:00:00 A	123 Cool St.	Buffalo	NY	14222	N	2	12/31/2021, 12:00:00 AM	11/9/2024, 4:47:46 AM	1
3	3	Sally Sallerson	3/3/1989, 12:00:00 A	415 Awesome Pl.	Rochester	NY	54321	Y	3	12/31/2021, 12:00:00 AM	12/31/2099, 12:00:00 AM	1
4	0	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	(null)	1
5	1000	Dominic Sellitto	(null)	123 New St.	Rochester	NY	14321	Y	1	11/9/2024, 4:47:45 AM	12/31/2199, 11:59:59 PM	2
6	1001	Jeep Jeeperson	(null)	123 Cool St.	Buffalo	NY	14043	Y	2	11/9/2024, 4:47:45 AM	12/31/2199, 11:59:59 PM	2
7	1002	James Bond	(null)	543 Bond Rd.	Buffalo	NY	14222	Y	4	11/9/2024, 4:47:45 AM	12/31/2199, 11:59:59 PM	1
8	1003	Jennifer Lopez	(null)	91 Perfect Ave.	Rochester	NY	14321	Y	5	11/9/2024, 4:47:45 AM	12/31/2199, 11:59:59 PM	1

Figure 14 Dim_Customer Updated via Apache hop flow

15. Pipeline 4-Fact Sales

In this workflow, we created a new pipeline in Apache Hop called "LOAD_FACT_SALES_STAGING" to load data into a fact table in the data warehouse. This process involved configuring multiple lookups for each foreign key, using a combination of CSV inputs and table lookups to match business keys with their corresponding surrogate keys from dimension tables. Through this setup, we used "Stream Lookup" nodes to retrieve current keys, ensuring accurate foreign key references in the fact table. This method streamlines data integration by maintaining continuous data flow and allowing efficient key matching across dimension and fact tables.

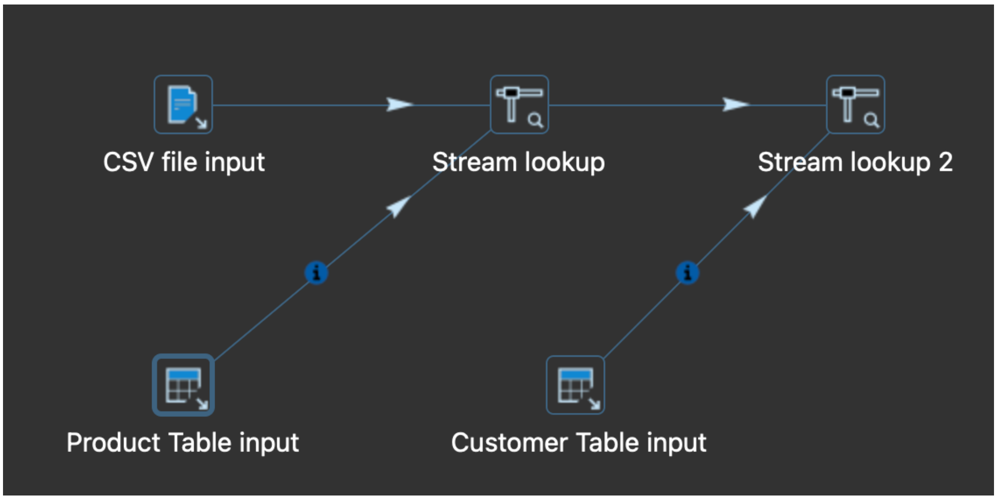
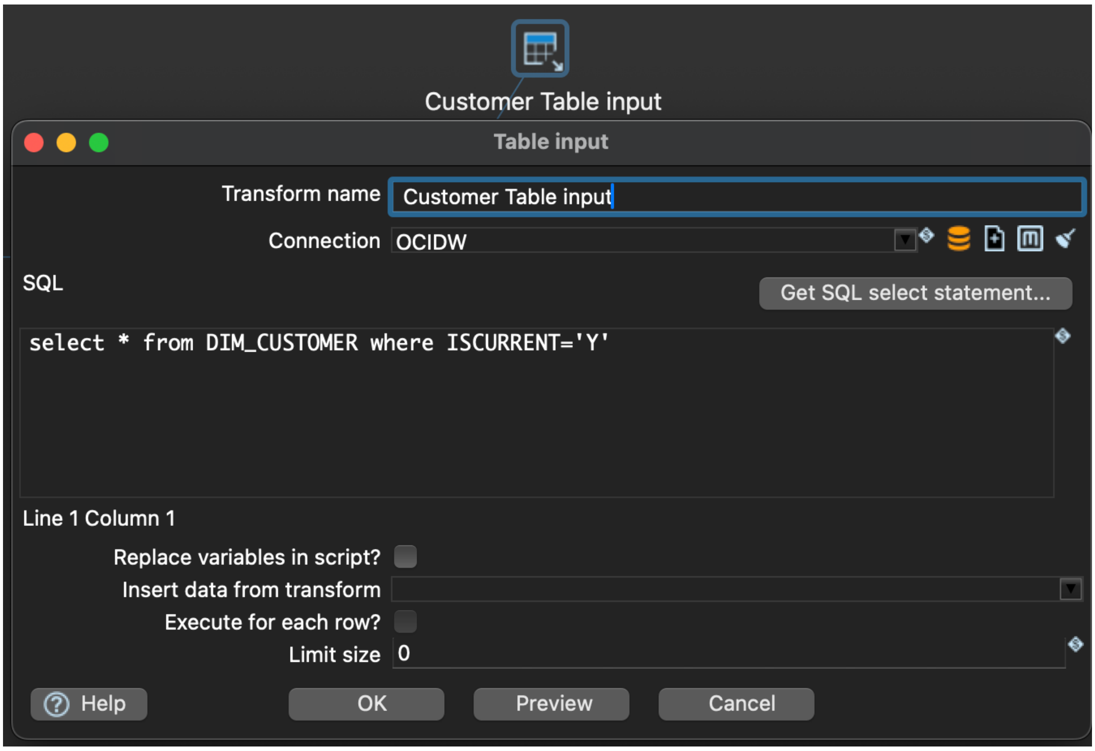


Figure 15 Pipeline 4-Fact Sales



Transform name: Customer Table input

Connection: OCIDW

SQL

```
select * from DIM_CUSTOMER where ISCURRENT='Y'
```

Line 1 Column 1

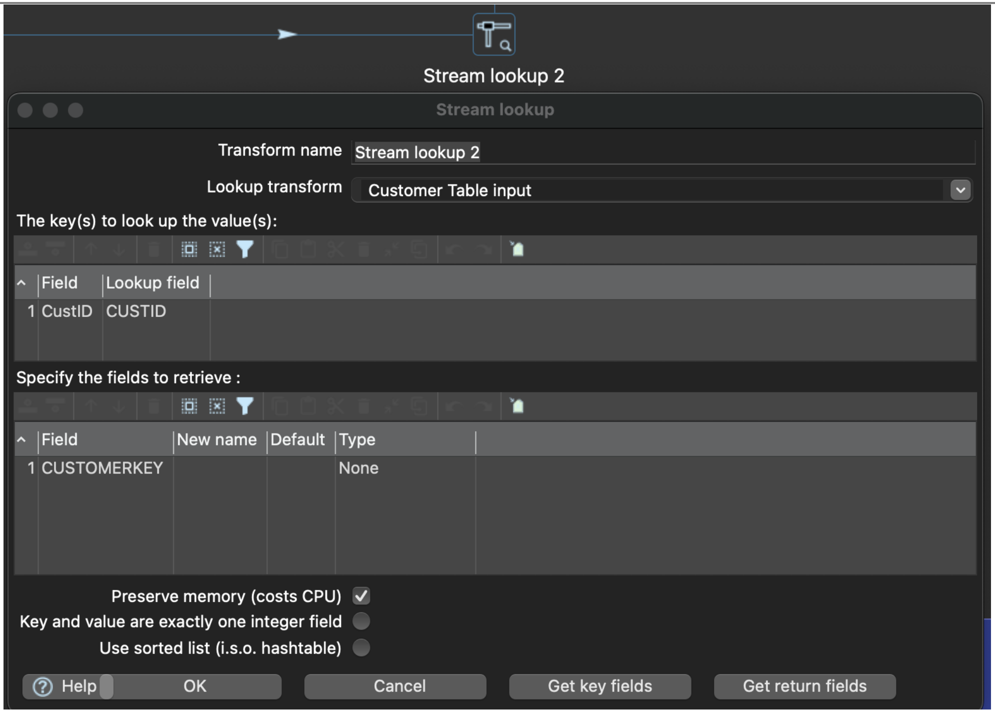
Replace variables in script? □

Insert data from transform

Execute for each row? □

Limit size: 0

Figure 16 Pipeline 4- Fact Sales(Customer Table Input)



Transform name: Stream lookup 2

Lookup transform: Customer Table input

The key(s) to look up the value(s):

	Field	Lookup field
1	CustID	CUSTID

Specify the fields to retrieve :

	Field	New name	Default	Type
1	CUSTOMERKEY			None

Preserve memory (costs CPU) ☑

Key and value are exactly one integer field ○

Use sorted list (i.s.o. hashtable) ○

Figure 17 Pipeline 4-Fact Sales(Customer Stream Lookup)

- You might have noticed we're not doing a lookup for the date dimension, why?

A lookup for the date dimension is unnecessary because:

- Unique DATEKEY: The DIM_DATE table already has a unique DATEKEY that identifies each date, allowing direct joins without requiring an additional surrogate key lookup.
- Direct Date Use in Fact Table: The fact table can use the date value directly, referencing DATEKEY as needed. Since the date information is static and consistent, pre-joining or merging with Dim_Date may not be necessary at every ETL run. This avoids redundancy and simplifies the join process with the date dimension.

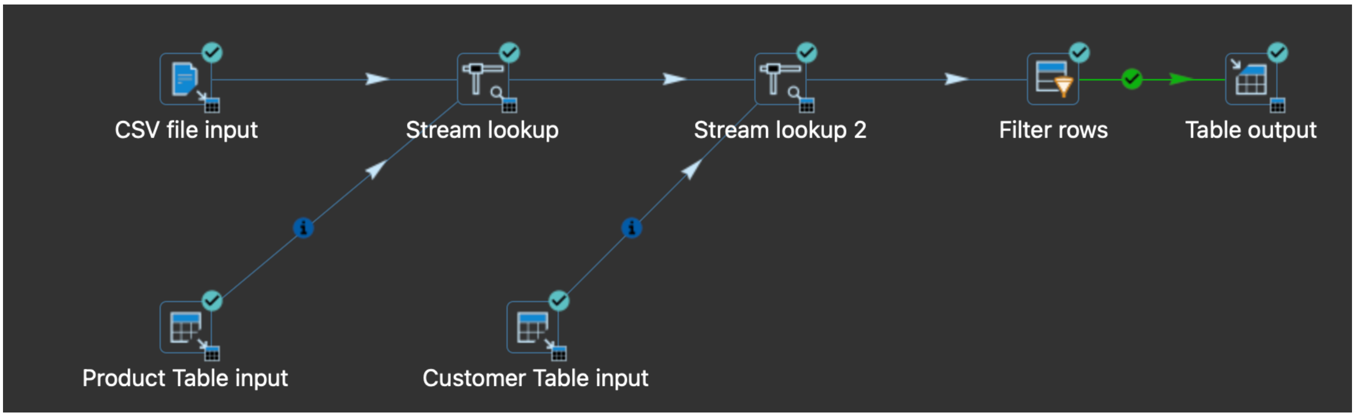
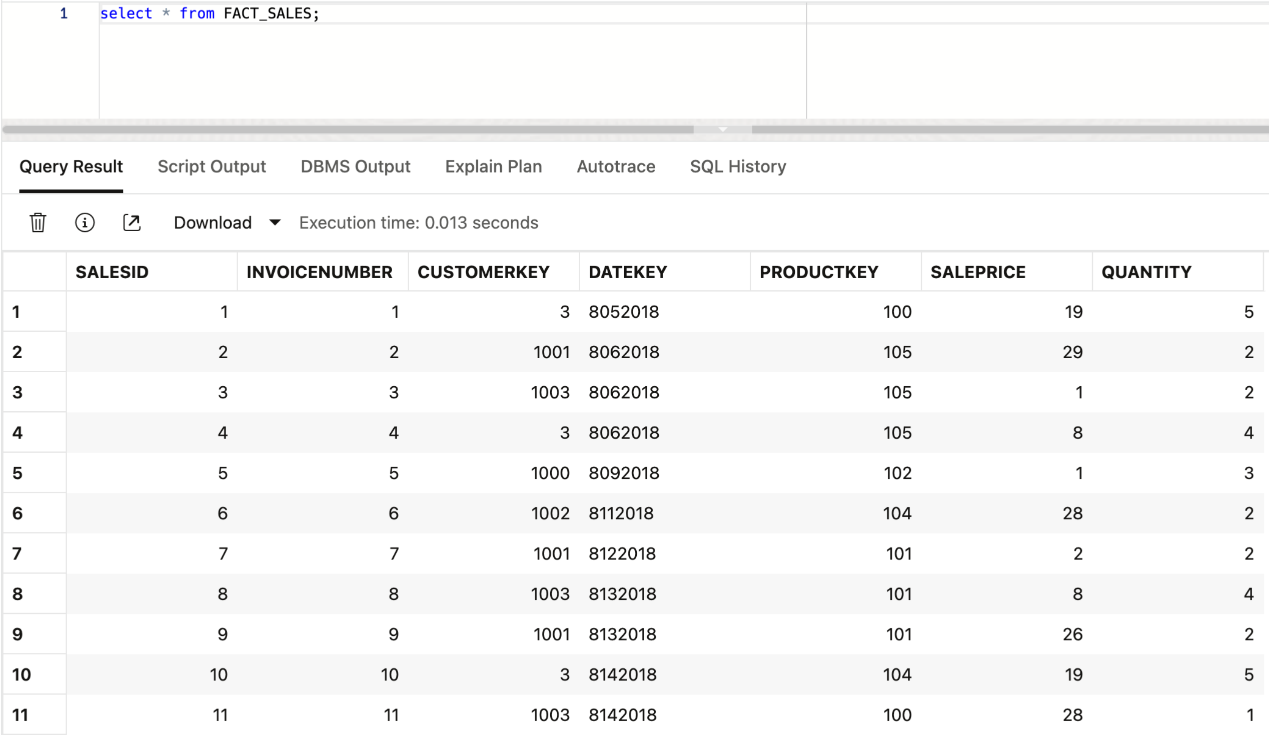


Figure 18 Pipeline 4-Fact Sales(Final Flow)

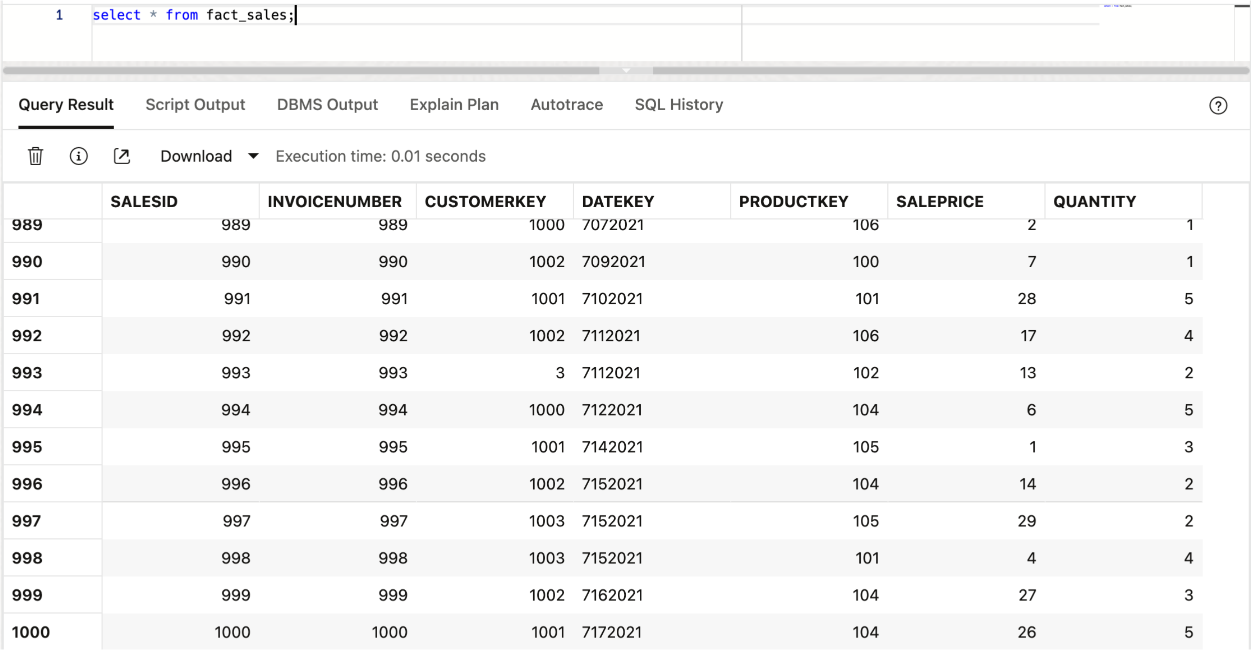


```
select * from FACT_SALES;
```

Execution time: 0.013 seconds

	SALESID	INVOICENUMBER	CUSTOMERKEY	DATEKEY	PRODUCTKEY	SALEPRICE	QUANTITY
1	1	1	3	8052018	100	19	5
2	2	2	1001	8062018	105	29	2
3	3	3	1003	8062018	105	1	2
4	4	4	3	8062018	105	8	4
5	5	5	1000	8092018	102	1	3
6	6	6	1002	8112018	104	28	2
7	7	7	1001	8122018	101	2	2
8	8	8	1003	8132018	101	8	4
9	9	9	1001	8132018	101	26	2
10	10	10	3	8142018	104	19	5
11	11	11	1003	8142018	100	28	1

Figure 19 Fact Sales(Head)



```
select * from fact_sales;
```

Execution time: 0.01 seconds

	SALESID	INVOICENUMBER	CUSTOMERKEY	DATEKEY	PRODUCTKEY	SALEPRICE	QUANTITY
989	989	989	1000	7072021	106	2	1
990	990	990	1002	7092021	100	7	1
991	991	991	1001	7102021	101	28	5
992	992	992	1002	7112021	106	17	4
993	993	993	3	7112021	102	13	2
994	994	994	1000	7122021	104	6	5
995	995	995	1001	7142021	105	1	3
996	996	996	1002	7152021	104	14	2
997	997	997	1003	7152021	105	29	2
998	998	998	1003	7152021	101	4	4
999	999	999	1002	7162021	104	27	3
1000	1000	1000	1001	7172021	104	26	5

Figure 20 Fact Sales(tail)