The Guac Shop
Creating a simple ETL pipeline

The project included many activities, all leading to the progressive creation of the ETL pipeline.

The project began with creating dimension and fact tables, including **Product, Date, Customer,** and **Sales**. These tables were essential for structuring the data, with initial data inserts carried out for the **Product, Sales,** and **Date** tables to establish a reliable dataset for pipeline building.

Following the data structuring, the setup phase included installing **Apache Hop**, **Java**, **JDBC**, and **Wallet**—each of which is critical for pipeline functionality. Apache Hop served as the primary ETL tool, enabling smooth data flow and transformation, while JDBC facilitated database connectivity. With these installations in place, a connection was established between **Oracle Cloud** and **Apache Hop**, allowing the ETL tool to access and manipulate data stored in the cloud environment.

With infrastructure in place, the first ETL pipeline was developed, focusing on building dimensional loader. Following this, a **Slowly Changing Dimension (SCD)** pipeline was implemented for the **Customer** and **Product** tables, individually, to handle evolving data while maintaining historical accuracy—key for applications requiring detailed audit trails.

The final stage was the construction of a comprehensive ETL pipeline capable of integrating all dimensions and loading data into the **fact tables**. This complete pipeline enabled a fullscale data integration process, moving clean, organized data from source to destination for future analysis.

This project presented several challenges, particularly in ensuring accurate connectivity between Oracle Cloud and Apache Hop and configuring the SCD pipeline to handle data

updates effectively. The project underscored the importance of a structured approach to ETL development, highlighting each step's role in building a scalable, reliable data pipeline.

At the end, the ETL pipeline was successfully developed, allowing for efficient data handling and transformation. This process provided valuable insights into the complexities of ETL pipeline creation, from initial setup to the final integration, and underscored the importance of rigorous testing and configuration in delivering a fully functional data management system.

Image 1: Diagram of Database

The diagram is missing relationships because foreign key constraints are not in place. Furthermore, the Customer and Product tables lack primary key tagging, and the Fact Sales table does not contain foreign keys, preventing accurate mapping.

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DAY_NUM_OF_WE	DAY_NUM_OF_MC	DAY_NUM_OF_YE/	MONTH_ID	MONTH_TIME_SPAN	MONTH_END_DA1	MONTH_SHORT_DESC	MONTH_LONG_DI	MON
1	20180101	1/1/2018, 12:00:00 A	1	1/1/2018, 12:00:00 A	Monday	MON	2	1	1	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
2	20180102	1/2/2018, 12:00:00 A	1	1/2/2018, 12:00:00 A	Tuesday	TUE	3	2	2	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
3	20180103	1/3/2018, 12:00:00 A	1	1/3/2018, 12:00:00 A	Wednesday	WED	4	3	3	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
4	20180104	1/4/2018, 12:00:00 A	1	1/4/2018, 12:00:00 A	Thursday	THU	5	4	4	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
5	20180105	1/5/2018, 12:00:00 A	1	1/5/2018, 12:00:00 A	Friday	FRI	6	5	5	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
6	20180106	1/6/2018, 12:00:00 A	1	1/6/2018, 12:00:00 A	Saturday	SAT	7	6	6	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
7	20180107	1/7/2018, 12:00:00 A	1	1/7/2018, 12:00:00 A	Sunday	SUN	1	7	7	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
8	20180108	1/8/2018, 12:00:00 A	1	1/8/2018, 12:00:00 A	Monday	MON	2	8	8	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
9	20180109	1/9/2018, 12:00:00 A	1	1/9/2018, 12:00:00 A	Tuesday	TUE	3	9	9	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
10	20180110	1/10/2018, 12:00:00 .	1	1/10/2018, 12:00:00 /	Wednesday	WED	4	10	10	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
11	20180111	1/11/2018, 12:00:00 /	1	1/11/2018, 12:00:00 /	Thursday	THU	5	11	11	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan
12	20180112	1/12/2018, 12:00:00 .	1	1/12/2018, 12:00:00 /	Friday	FRI	6	12	12	JAN-2018	31	1/31/2026, 12:00:00	Jan 2018	January 2018	Jan

*Image 2: Dim_Date Output **(Before)***

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DAY_NUM_OF_WE	DAY_NUM_OF_MO	DAY_NUM_OF_YE	MONTH_ID	MONTH_TIME_SPAN	MONTH_END_DAT	MONTH_SHORT_DESC	MONTH_LONG_DE	MON
1	20160101	1/1/2016, 12:00:00 A	1	1/1/2016, 12:00:00 A	Friday	FRI	6	1	1	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
2	20160102	1/2/2016, 12:00:00 A	1	1/2/2016, 12:00:00 A	Saturday	SAT	7	2	2	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
3	20160103	1/3/2016, 12:00:00 A	1	1/3/2016, 12:00:00 A	Sunday	SUN	1	3	3	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
4	20160104	1/4/2016, 12:00:00 A	1	1/4/2016, 12:00:00 A	Monday	MON	2	4	4	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
5	20160105	1/5/2016, 12:00:00 A	1	1/5/2016, 12:00:00 A	Tuesday	TUE	3	5	5	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
6	20160106	1/6/2016, 12:00:00 A	1	1/6/2016, 12:00:00 A	Wednesday	WED	4	6	6	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
7	20160107	1/7/2016, 12:00:00 A	1	1/7/2016, 12:00:00 A	Thursday	THU	5	7	7	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
8	20160108	1/8/2016, 12:00:00 A	1	1/8/2016, 12:00:00 A	Friday	FRI	6	8	8	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
9	20160109	1/9/2016, 12:00:00 A	1	1/9/2016, 12:00:00 A	Saturday	SAT	7	9	9	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
10	20160110	1/10/2016, 12:00:00 .	1	1/10/2016, 12:00:00 .	Sunday	SUN	1	10	10	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
11	20160111	1/11/2016, 12:00:00 /	1	1/11/2016, 12:00:00 /	Monday	MON	2	11	11	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan
12	20160112	1/12/2016, 12:00:00 .	1	1/12/2016, 12:00:00 .	Tuesday	TUE	3	12	12	JAN-2016	31	1/31/2026, 12:00:00	Jan 2016	January 2016	Jan

*Image 3: Dim_Date Output (**After**)*

SQL Code for Dim Date

```
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,

-- 31 AS Month_Time_Span,

MAX(TO_NUMBER(TO_CHAR(CurrDate, 'DD'))) OVER (PARTITION BY TO_CHAR(CurrDate,'Mon')) AS
Month_Time_Span,

--to_date('31-JAN-2010','DD-MON-YYYY') AS Month_End_Date,

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,

-- 31 AS Quarter_Time_Span,
```

```
COUNT(*) OVER (PARTITION BY TO_CHAR(CurrDate,'Q')) AS Quarter_Time_Span,
-- to_date('31-JAN-2010','DD-MON-YYYY') AS Quarter_End_Date,
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,
--31 AS Year_Time_Span,
COUNT(*) OVER (PARTITION BY TO_CHAR(CurrDate,'YYYY')) AS Year_Time_Span, --
to_date('31-JAN-2010','DD-MON-YYYY') AS Year_End_Date
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);
```

SQL Code

DimDate Create and Populate.txt

The Guac Shop

Creating a simple ETL pipeline

Building First Dimensional Loader

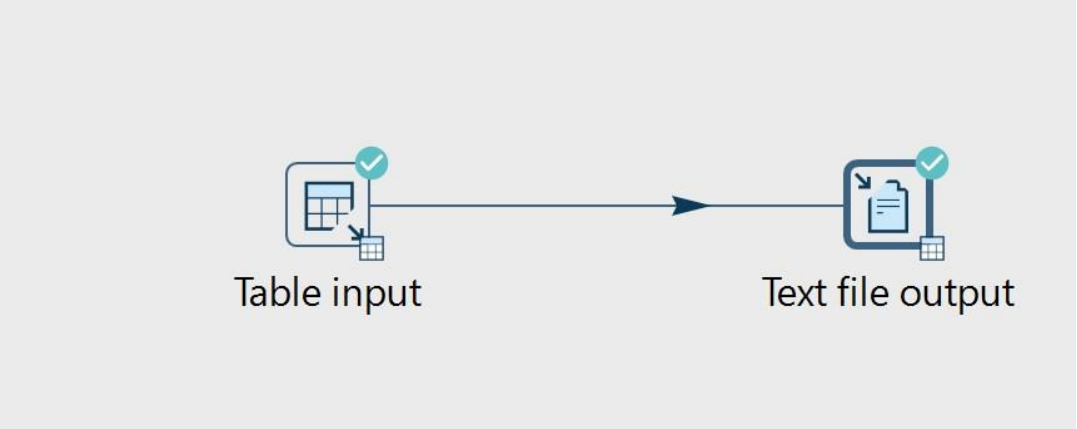

Image 4: Initial Pipeline

Output .txt file

test.txt

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

Image 5: Dim Product (Before)

	PRODUCTKEY	PRODUCTNAME	CATEGORY	SUBCATEGORY	BRAND	ISCURRENT	PRODUCTID	SCD_START	SCD_END	VERSIONING
1	0	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	1
2	1	Cinnamon Bread Loa	Wheat	Bread	Nothing Breader	Y	1	1/1/2024, 12:00:00 AM	12/31/2099, 12:00:00 AM	1
3	2	Milk	Dairy	Liquid	Buffalo Farms	Y	2	2/1/2024, 12:00:00 AM	12/31/2099, 12:00:00 AM	1
4	3	Chocolate Chip Cool	Candy	Cookies	Nothing Breader	Y	3	3/1/2024, 12:00:00 AM	12/31/2099, 12:00:00 AM	1
5	4	Eggs	Dairy	Solid	Rochester Farms	N	4	4/1/2024, 12:00:00 AM	11/15/2024, 8:01:11 PM	1
6	5	Rotini	Wheat	Pasta	Buffalo Farms	Y	5	5/1/2024, 12:00:00 AM	12/31/2099, 12:00:00 AM	1
7	100	Eggs	Poultry	Solid	Rochester Farms	Y	4	11/15/2024, 8:01:08 PM	12/31/2199, 11:59:59 PM	2
8	101	Sugary Cereal	Wheat	Cereal	Food For You	Y	6	11/15/2024, 8:01:08 PM	12/31/2199, 11:59:59 PM	1

Image 6: Dim Product (After)

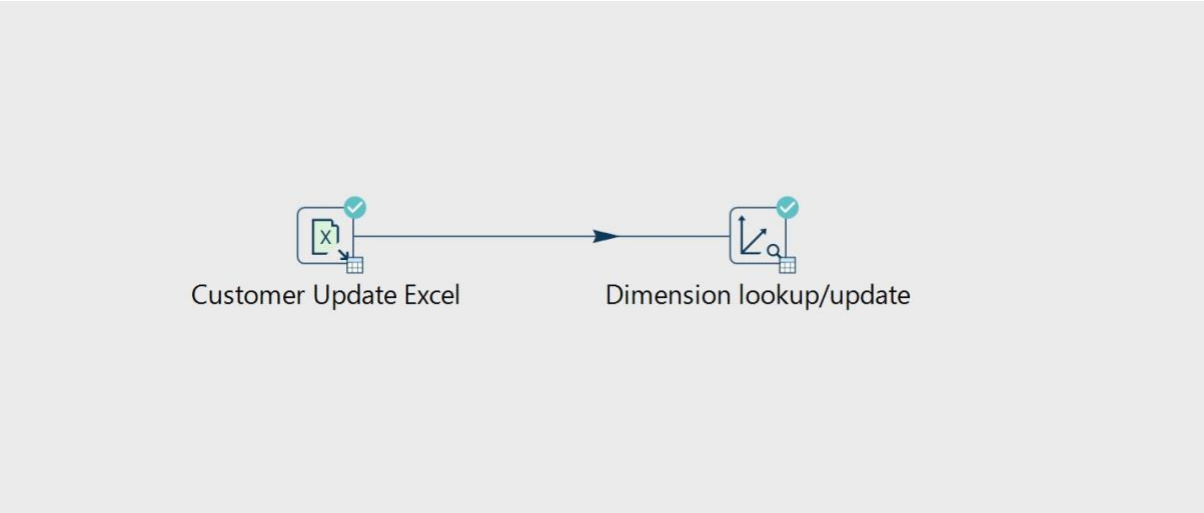

Image 7: Dim Customer Pipeline

	CUSTOMERKEY	CNAME	BIRTHDAY	CADDRESS	CITY	STATEPROV	ZIP	ISCURRENT	CUSTID	SCD_START	SCD_END	VERSIONING
1	0	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	1
2	1	Dominic Sellitto	1/1/1956, 12:00:00 AM	123 ABC St.	Buffalo	NY	14222	N	1	12/31/2021, 12:00:00 AM	11/15/2024, 8:11:03 PM	1
3	2	Jeep Jeeperson	2/2/1979, 12:00:00 AM	123 Cool St.	Buffalo	NY	14222	N	2	12/31/2021, 12:00:00 AM	11/15/2024, 8:11:03 PM	1
4	3	Sally Sallerson	3/3/1989, 12:00:00 AM	415 Awesome Pl.	Rochester	NY	54321	Y	3	12/31/2021, 12:00:00 AM	12/31/2099, 12:00:00 AM	1
5	1000	Dominic Sellitto	*(null)*	123 New St.	Rochester	NY	14321	Y	1	11/15/2024, 8:11:00 PM	12/31/2199, 11:59:59 PM	2
6	1001	Jeep Jeeperson	*(null)*	123 Cool St.	Buffalo	NY	14043	Y	2	11/15/2024, 8:11:00 PM	12/31/2199, 11:59:59 PM	2
7	1002	James Bond	*(null)*	543 Bond Rd.	Buffalo	NY	14222	Y	4	11/15/2024, 8:11:00 PM	12/31/2199, 11:59:59 PM	1
8	1003	Jennifer Lopez	*(null)*	91 Perfect Ave.	Rochester	NY	14321	Y	5	11/15/2024, 8:11:00 PM	12/31/2199, 11:59:59 PM	1

Image 8: Dim Customer (After)

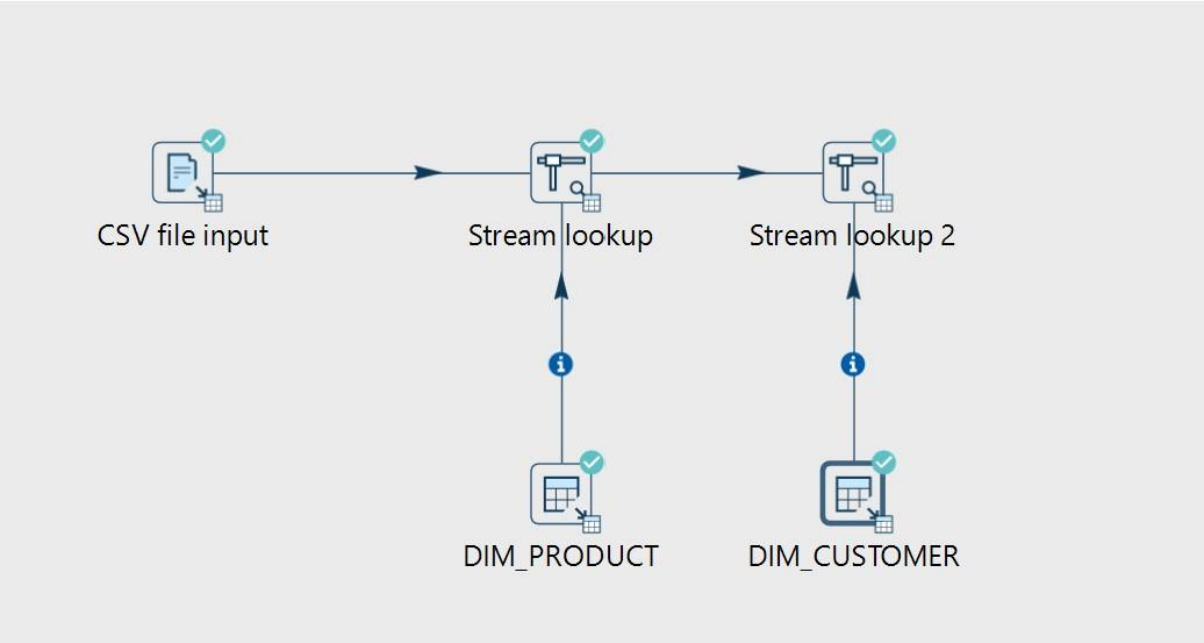

Image 9: Stream Lookup 2

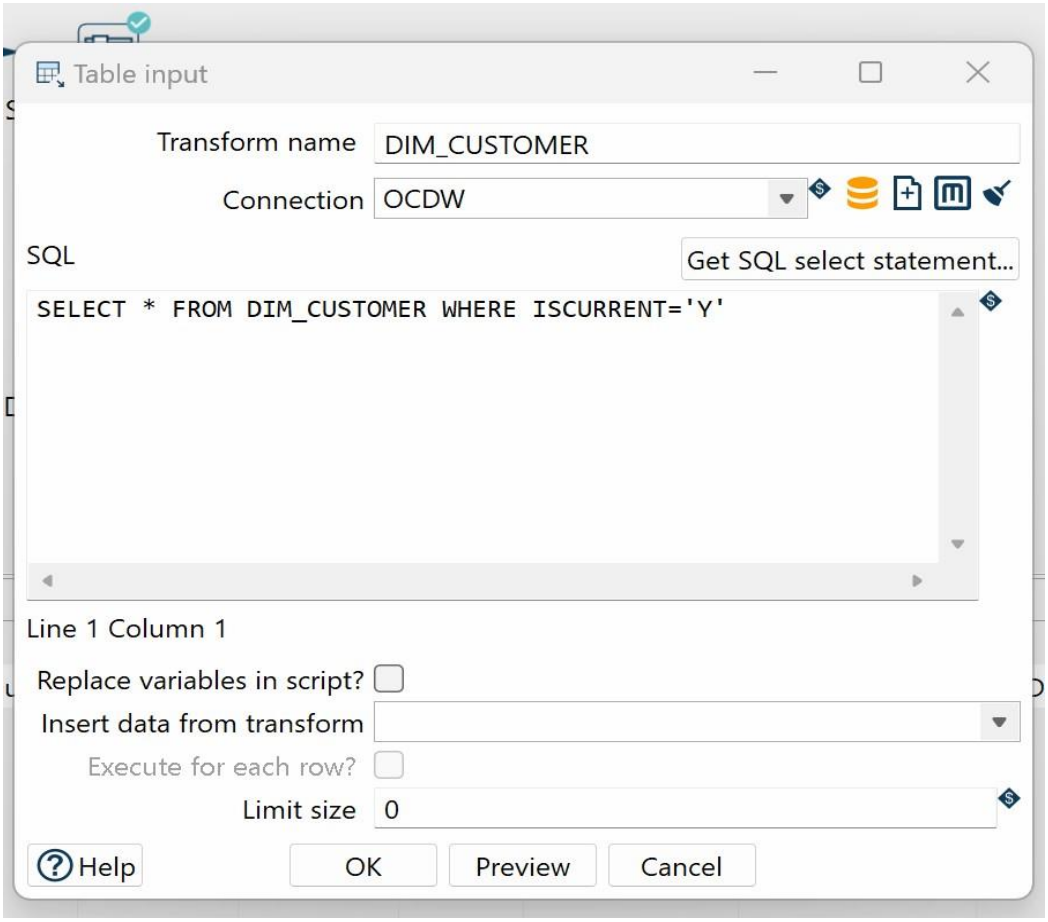

Image 10: Stream Lookup 2 Input Table (edit mode)

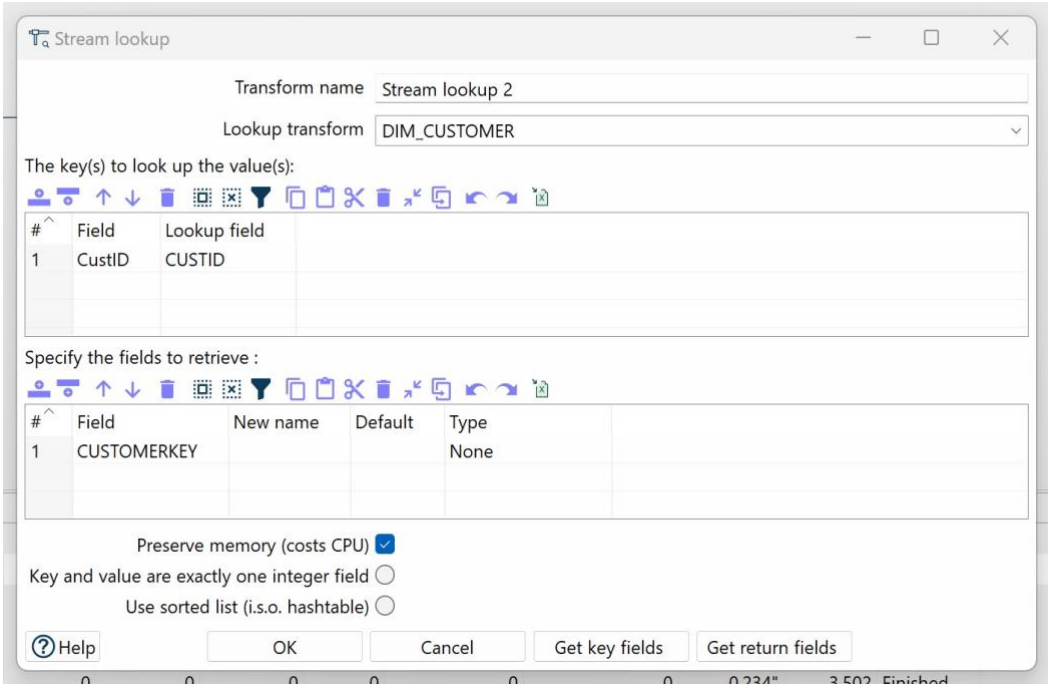

Image 11: Stream Lookup 2 (edit mode)

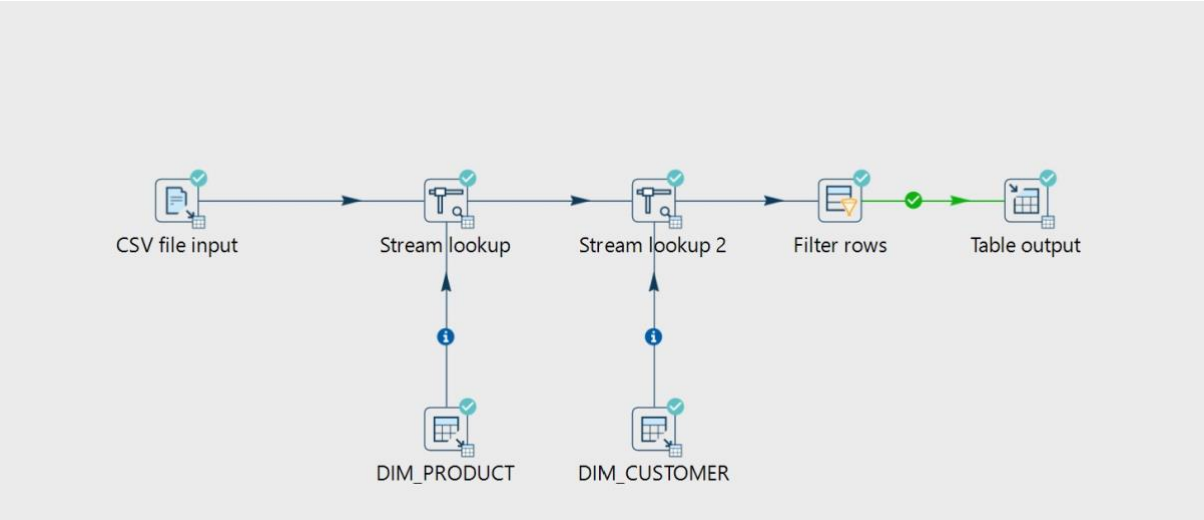

Image 12: Final Pipeline

The date dimension table is not used because the date key in **sales_fact** has a different format. Additionally, the date field is unique and self-contained, so a lookup is unnecessary.

	SALESID	INVOICENUMBER	CUSTOMERKEY	DATEKEY	PRODUCTKEY	SALEPRICE	QUANTITY
1	1	1	3	8052018	1	19	5
2	2	2	1001	8062018	100	29	2
3	3	3	1003	8062018	100	1	2
4	4	4	3	8062018	100	8	4
5	5	5	1000	8092018	3	1	3
6	6	6	1002	8112018	5	28	2
7	7	7	1001	8122018	2	2	2
8	8	8	1003	8132018	2	8	4
9	9	9	1001	8132018	2	26	2
10	10	10	3	8142018	5	19	5
11	11	11	1003	8142018	1	28	1
12	12	12	1001	8162018	101	30	1
13	13	13	1002	8212018	5	15	5
14	14	14	1003	8232018	100	18	4

Image 13: Final Output

Final_Output.csv

Final_Output.xlsx