Homework / Lab 1

Proposing a Data Warehouse

MGS 6577LEC F1S: Cloud Data Warehousing & Data Engn (19774 Fall24)

Submitted By

Anjali Pandey

Contents

Summary of ETL Implementation Activities

This assignment main goal was to implement an Extract, Transform, Load (ETL) process for populating a data warehouse hosted on Oracle Cloud. With the use of Apache Hop, I built and tested pipelines to automated data loading and processing tasks. Below are the steps which I followed in completion of this assignment.

Key Activities:

1. Oracle Cloud Setup:

- With the use of Autonomous Datawarehouse(ADW), existing tables were dropped and recreated new tables using the provided DDL scripts.
- The diagram was generated post-DDL execution, highlighting missing feature and their implications.

2. Populating Dimension table:

- Executed scripts of DIM_DATE table with a specified date range (2016-2026), updating parameters as per requirements.
- Insertion of records were done manually into DIM_CUSTOMER and DIM_PRODUCT table, maintain correct format and constraints.

3. Apache Hop Installation:

- Apache Hop v29.0 was installed, and Java dependencies were set up for seamless operations.
- JDBC configuration and wallet configuration was done and linked to Apache Hop with Oracle Cloud ADW. The manual construction of connection strings involved precise replacement of placeholders with database-specific values.

4. Building ETL Pipelines:

- Initially a foundation pipeline was extracted from DIM_CUSTOMER and transformed them into a text file. This helped in hands-on with basic ETL workflow.
- A whole dedicated workflow was built to handle slowly changing dimensions in DIM_PRODUCT table:
    - Type 1 Updates: ProductName attributes were updated directly for tracking changes .
    - Type 2 Inserts: Tracking was enabled for attributes like Category and Subcategory.
    - ISCURRENT Field Automation: The workflow leveraged Apache Hop's 'Last version' feature for managing active records.
- Field Mapping and QA: Fields from input data were mapped to dimension table columns, ensuring accurate updates. SQL queries validated the integrity of changes post-execution.

5. Fact Table Loading:

- Then the pipeline loaded data into FACT_SALES tables, performing multiple stream lookups:

    - **Foreign Key Resolution:** Business keys from the CSV input (e.g., ProductID, CustomerID) were mapped to surrogate keys (ProductKey, CustomerKey) via dimension table queries.
    - **Early Fact Filtering**: Null values in surrogate keys were filtered out, preventing incomplete records from entering the warehouse.
- **Pipeline Execution:** Filtered data was loaded into FACT_SALES using a Table Output node, ensuring schema compliance.

Additional Findings:

- **Pipeline Versatility**: Apache Hop's flexibility allowed for handling diverse data sources and transformations. Its user-friendly GUI enabled seamless pipeline adjustments.
- **Data Warehouse Challenges**: Missing database constraints and indexes highlighted areas for further optimization. The importance of maintaining accurate metadata during ETL processes was emphasized.
- **Learning Outcomes**: The assignment demonstrated how to integrate an on-premise ETL tool with a cloud-based warehouse, offering a robust framework for scalable data processing.

Challenges Encountered:

- **Technical Configuration**: Aligning wallet files with the database connection and resolving Java dependency issues required iterative troubleshooting.
- **Memory Management**: Stream lookups occasionally caused high memory usage in Apache Hop, necessitating the use of optimized node configurations.
- **Manual Data Entry Errors:** Precision was critical in SQL insert statements, as small formatting issues led to failures during execution.

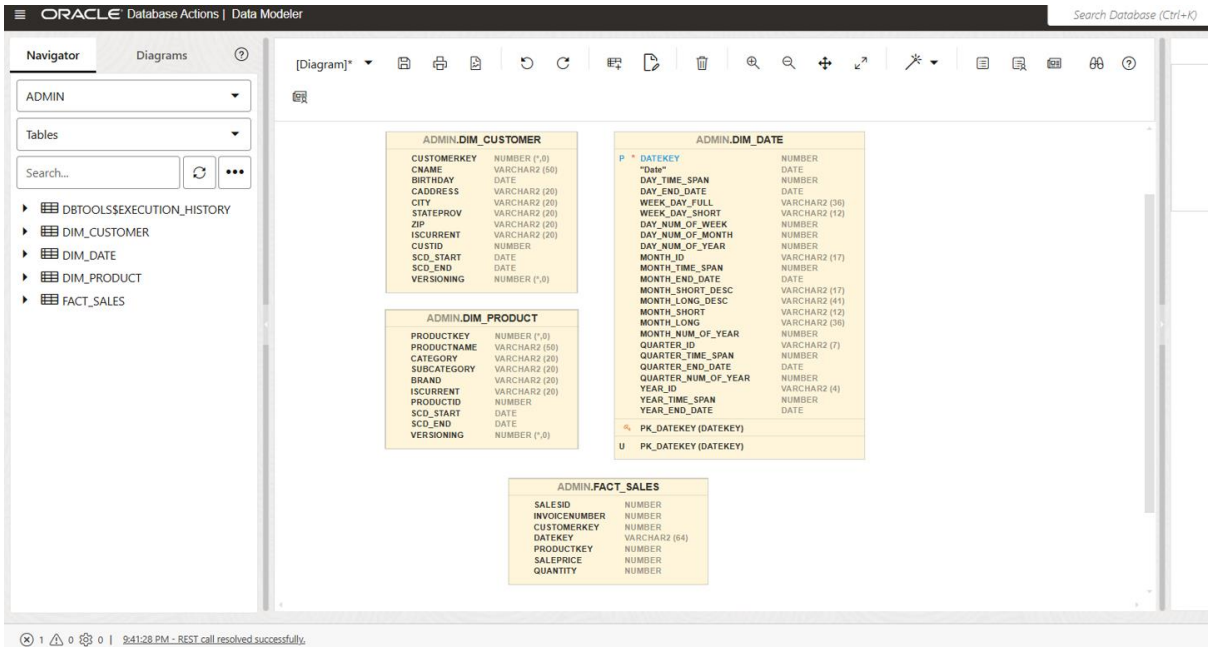
Conclusion:

This assignment demonstrated the end-to-end process of setting up and automating ETL tasks for data warehousing. Apache Hop proved effective for building flexible pipelines. The integration of Oracle Cloud with ETL processes highlighted key considerations for production-scale implementations. These activities have laid the groundwork for robust, scalable data management solutions.

Appendix

Introduction

1. *Database Design*

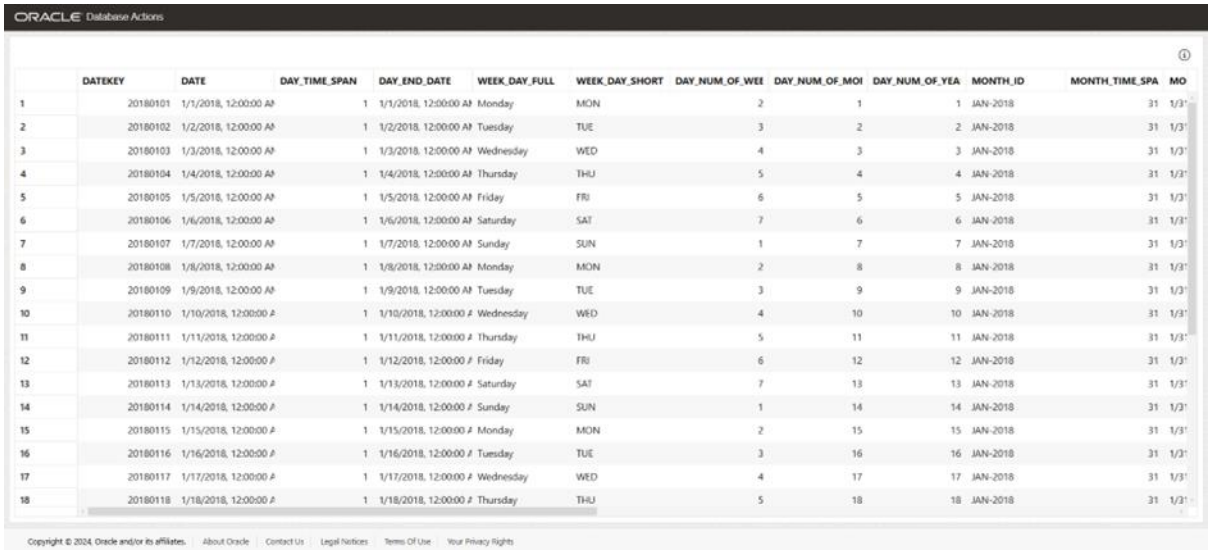


2. *Based on the diagram generated, what is this database missing that you'd expect to see? Why might it be missing this component?*

Ans - The database lacks Foreign Key and Primary Key relationships, which are typically used to maintain referential integrity. This absence is intentional, as referential integrity is being managed within the ETL pipeline instead. Including these relationships in the database would impact ETL performance, as the system would need to validate constraints with each record load. By shifting this responsibility to the ETL pipeline, the database is optimized for analytical tasks, reducing non-analytical overhead.

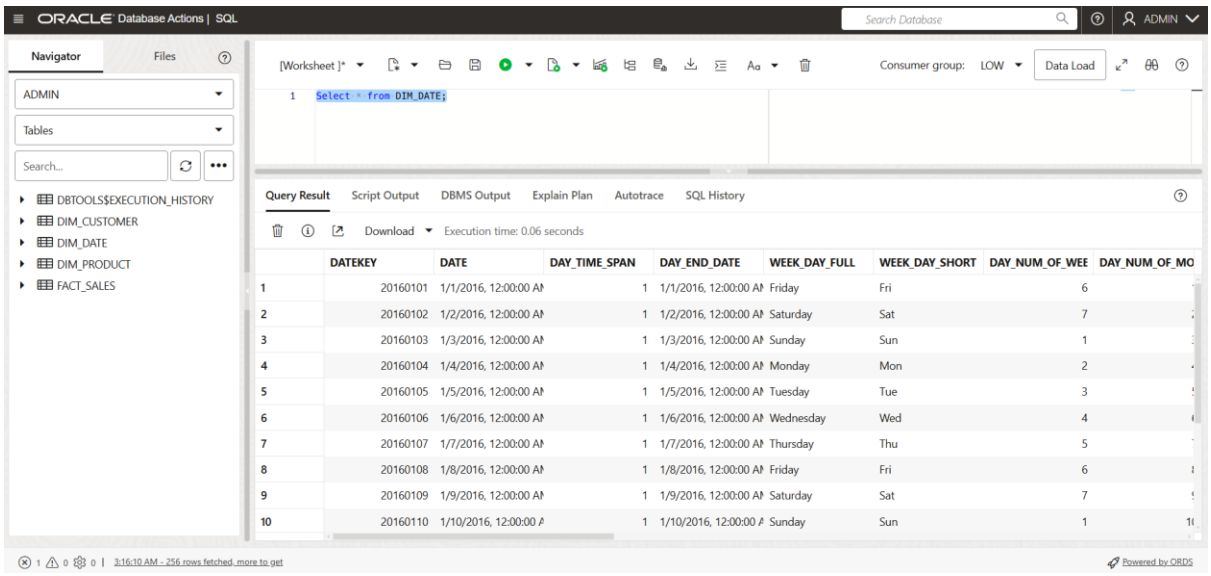
Loading Dimension Tables

1) *DIM_DATE initial screen snip*

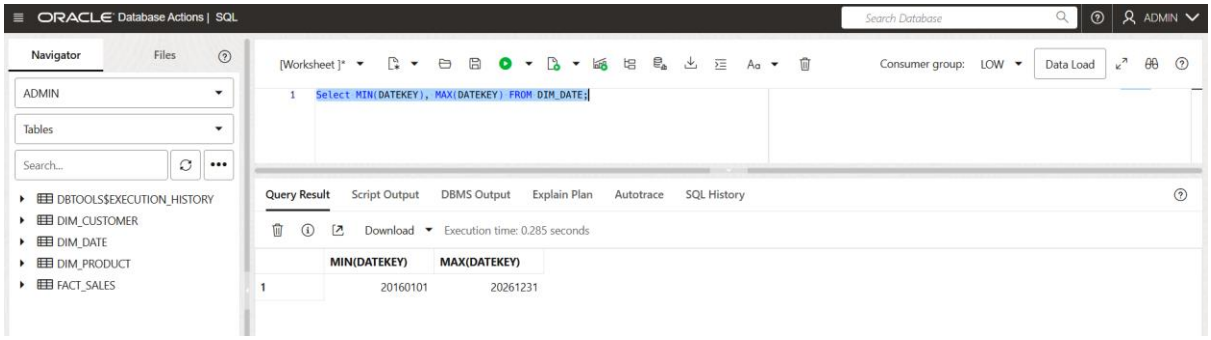


	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHORT	DAY_NUM_OF_WEE	DAY_NUM_OF_MOI	DAY_NUM_OF_YEA	MONTH_ID	MONTH_TIME_SPA	MO
1	20180101	1/1/2018, 12:00:00 AM	1	1/1/2018, 12:00:00 AM	Monday	MON	2	1	1	JAN-2018	31	1/3
2	20180102	1/2/2018, 12:00:00 AM	1	1/2/2018, 12:00:00 AM	Tuesday	TUE	3	2	2	JAN-2018	31	1/3
3	20180103	1/3/2018, 12:00:00 AM	1	1/3/2018, 12:00:00 AM	Wednesday	WED	4	3	3	JAN-2018	31	1/3
4	20180104	1/4/2018, 12:00:00 AM	1	1/4/2018, 12:00:00 AM	Thursday	THU	5	4	4	JAN-2018	31	1/3
5	20180105	1/5/2018, 12:00:00 AM	1	1/5/2018, 12:00:00 AM	Friday	FRI	6	5	5	JAN-2018	31	1/3
6	20180106	1/6/2018, 12:00:00 AM	1	1/6/2018, 12:00:00 AM	Saturday	SAT	7	6	6	JAN-2018	31	1/3
7	20180107	1/7/2018, 12:00:00 AM	1	1/7/2018, 12:00:00 AM	Sunday	SUN	1	7	7	JAN-2018	31	1/3
8	20180108	1/8/2018, 12:00:00 AM	1	1/8/2018, 12:00:00 AM	Monday	MON	2	8	8	JAN-2018	31	1/3
9	20180109	1/9/2018, 12:00:00 AM	1	1/9/2018, 12:00:00 AM	Tuesday	TUE	3	9	9	JAN-2018	31	1/3
10	20180110	1/10/2018, 12:00:00 A	1	1/10/2018, 12:00:00 A	Wednesday	WED	4	10	10	JAN-2018	31	1/3
11	20180111	1/11/2018, 12:00:00 A	1	1/11/2018, 12:00:00 A	Thursday	THU	5	11	11	JAN-2018	31	1/3
12	20180112	1/12/2018, 12:00:00 A	1	1/12/2018, 12:00:00 A	Friday	FRI	6	12	12	JAN-2018	31	1/3
13	20180113	1/13/2018, 12:00:00 A	1	1/13/2018, 12:00:00 A	Saturday	SAT	7	13	13	JAN-2018	31	1/3
14	20180114	1/14/2018, 12:00:00 A	1	1/14/2018, 12:00:00 A	Sunday	SUN	1	14	14	JAN-2018	31	1/3
15	20180115	1/15/2018, 12:00:00 A	1	1/15/2018, 12:00:00 A	Monday	MON	2	15	15	JAN-2018	31	1/3
16	20180116	1/16/2018, 12:00:00 A	1	1/16/2018, 12:00:00 A	Tuesday	TUE	3	16	16	JAN-2018	31	1/3
17	20180117	1/17/2018, 12:00:00 A	1	1/17/2018, 12:00:00 A	Wednesday	WED	4	17	17	JAN-2018	31	1/3
18	20180118	1/18/2018, 12:00:00 A	1	1/18/2018, 12:00:00 A	Thursday	THU	5	18	18	JAN-2018	31	1/3

2) *DIM_DATE screen snip after update*



	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHORT	DAY_NUM_OF_WEE	DAY_NUM_OF_MO
1	20160101	1/1/2016, 12:00:00 AM	1	1/1/2016, 12:00:00 AM	Friday	Fri	6	
2	20160102	1/2/2016, 12:00:00 AM	1	1/2/2016, 12:00:00 AM	Saturday	Sat	7	
3	20160103	1/3/2016, 12:00:00 AM	1	1/3/2016, 12:00:00 AM	Sunday	Sun	1	
4	20160104	1/4/2016, 12:00:00 AM	1	1/4/2016, 12:00:00 AM	Monday	Mon	2	
5	20160105	1/5/2016, 12:00:00 AM	1	1/5/2016, 12:00:00 AM	Tuesday	Tue	3	
6	20160106	1/6/2016, 12:00:00 AM	1	1/6/2016, 12:00:00 AM	Wednesday	Wed	4	
7	20160107	1/7/2016, 12:00:00 AM	1	1/7/2016, 12:00:00 AM	Thursday	Thu	5	
8	20160108	1/8/2016, 12:00:00 AM	1	1/8/2016, 12:00:00 AM	Friday	Fri	6	
9	20160109	1/9/2016, 12:00:00 AM	1	1/9/2016, 12:00:00 AM	Saturday	Sat	7	
10	20160110	1/10/2016, 12:00:00 A	1	1/10/2016, 12:00:00 A	Sunday	Sun	1	



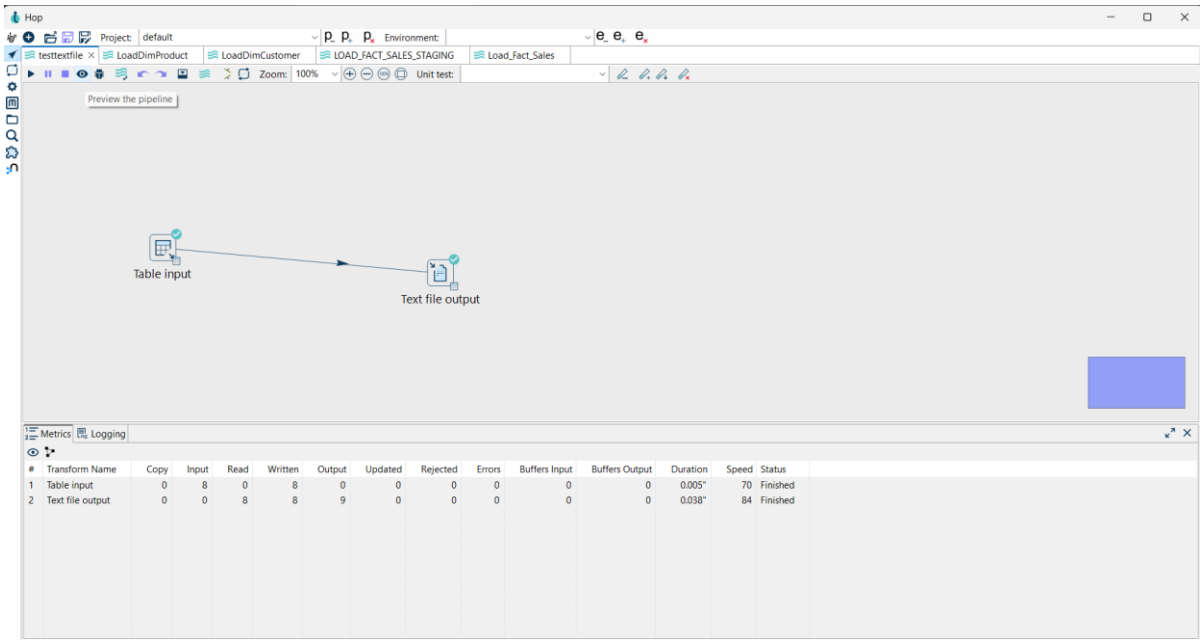
	MIN(DATEKEY)	MAX(DATEKEY)
1	20160101	20261231

3) *Updated script for date dimension*

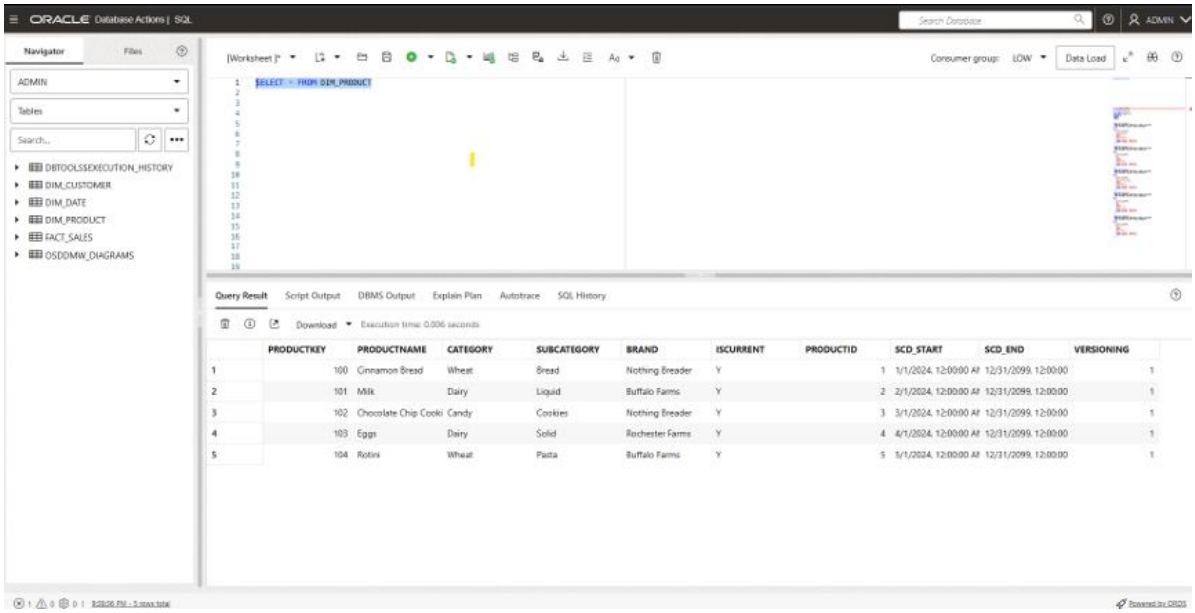
```
DROP TABLE IF EXISTS Dim_Date;

CREATE TABLE Dim_Date (
    DATEKEY INT PRIMARY KEY,
    "Date" DATE,
    DAY_TIME_SPAN INT,
    DAY_END_DATE DATE,
    WEEK_DAY_FULL VARCHAR(10),
    WEEK_DAY_SHORT VARCHAR(3),
    DAY_NUM_OF_WEEK INT,
    DAY_NUM_OF_MONTH INT,
    DAY_NUM_OF_YEAR INT,
    MONTH_ID VARCHAR(10),
    MONTH_TIME_SPAN INT,
    MONTH_END_DATE DATE,
    MONTH_SHORT_DESC VARCHAR(20),
    MONTH_LONG_DESC VARCHAR(20),
    MONTH_SHORT VARCHAR(3),
    MONTH_LONG VARCHAR(10),
    MONTH_NUM_OF_YEAR INT,
    QUARTER_ID VARCHAR(7),
    QUARTER_TIME_SPAN INT,
    QUARTER_END_DATE DATE,
    QUARTER_NUM_OF_YEAR INT,
    YEAR_ID INT,
    YEAR_TIME_SPAN INT,
    YEAR_END_DATE DATE
);

INSERT INTO Dim_Date
SELECT
    TO_NUMBER(TO_CHAR("Date", 'YYYYMMDD')) AS DATEKEY,
    "Date",
    1 AS DAY_TIME_SPAN,
    "Date" AS DAY_END_DATE,
    TO_CHAR("Date", 'Day') AS WEEK_DAY_FULL,
    TO_CHAR("Date", 'Dy') AS WEEK_DAY_SHORT,
    TO_NUMBER(TO_CHAR("Date", 'D')) AS DAY_NUM_OF_WEEK,
    TO_NUMBER(TO_CHAR("Date", 'DD')) AS DAY_NUM_OF_MONTH,
    TO_NUMBER(TO_CHAR("Date", 'DDD')) AS DAY_NUM_OF_YEAR,
    TO_CHAR("Date", 'MON-yyyy') AS MONTH_ID,
    MAX(TO_NUMBER(TO_CHAR("Date", 'DD'))) OVER (PARTITION BY
TO_CHAR("Date",'Mon')) AS MONTH_TIME_SPAN,
    MAX("Date") OVER (PARTITION BY TO_CHAR("Date",'Mon')) as
MONTH_END_DATE,
    TO_CHAR("Date",'Mon') || ' ' || TO_CHAR("Date",'YYYY') AS
MONTH_SHORT_DESC,
```

```
   RTRIM(TO_CHAR("Date",'Month')) || ' ' || TO_CHAR("Date",'YYYY') AS
MONTH_LONG_DESC,
   TO_CHAR("Date", 'MON') AS MONTH_SHORT,
   TO_CHAR("Date", 'MONTH') AS MONTH_LONG,
    TO_NUMBER(TO_CHAR("Date", 'MM')) AS MONTH_NUM_OF_YEAR,
    CONCAT('Q', TO_CHAR("Date", 'Q')) || '-' || TO_CHAR("Date", 'YYYY')
AS QUARTER_ID,
    800 AS QUARTER_TIME_SPAN,
    LAST_DAY(ADD_MONTHS("Date", 3 - MOD(TO_CHAR("Date", 'MM') - 1, 3)))
AS QUARTER_END_DATE,
    TO_NUMBER(TO_CHAR("Date", 'Q')) AS QUARTER_NUM_OF_YEAR,
    TO_NUMBER(TO_CHAR("Date", 'YYYY')) AS YEAR_ID,
    CASE
        WHEN MOD(TO_CHAR("Date", 'YYYY'), 4) = 0 AND TO_CHAR("Date",
'YYYY') != '1900' THEN 366
        ELSE 365
    END AS YEAR_TIME_SPAN,
    TO_DATE(TO_CHAR("Date", 'YYYY') || '1231', 'YYYYMMDD') AS
YEAR_END_DATE
FROM
    (SELECT TO_DATE('2016-01-01', 'YYYY-MM-DD') + LEVEL - 1 AS "Date"
     FROM dual
     CONNECT BY LEVEL <= (TO_DATE('2026-12-31', 'YYYY-MM-DD') -
TO_DATE('2016-01-01', 'YYYY-MM-DD')) + 1);
```

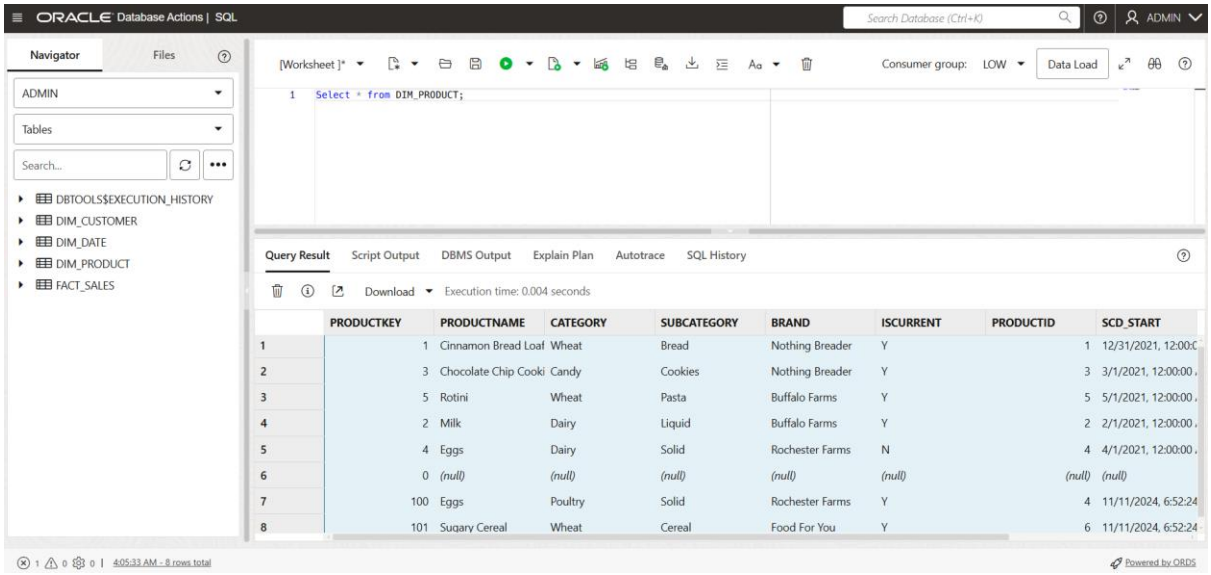
4) *Test Pipeline screen snip*



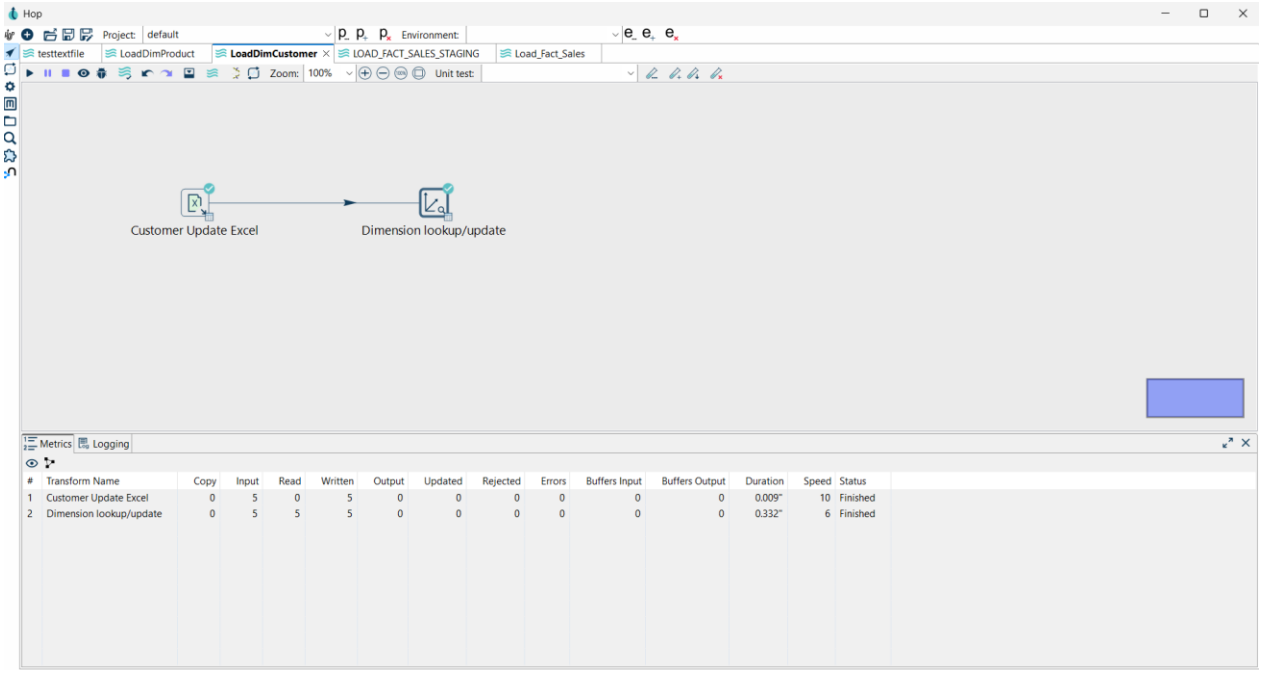
5) DIM_PRODUCT initial screen snip



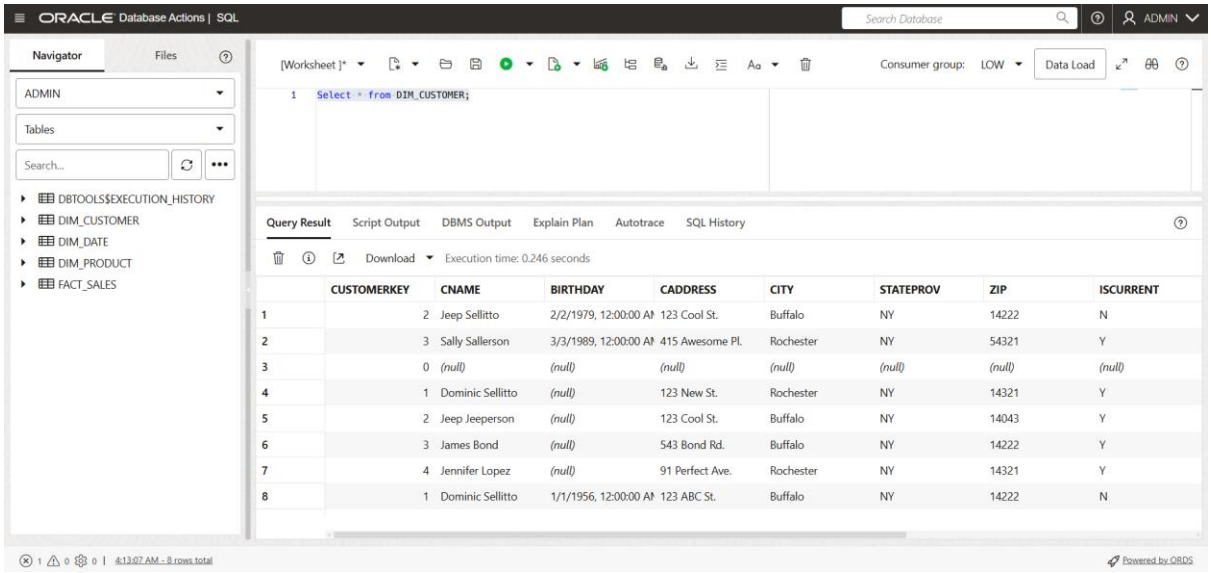
6) DIM_PRODUCT screen snip after ETL Pipeline execution



7) DIM_CUSTOMER ETL Pipeline

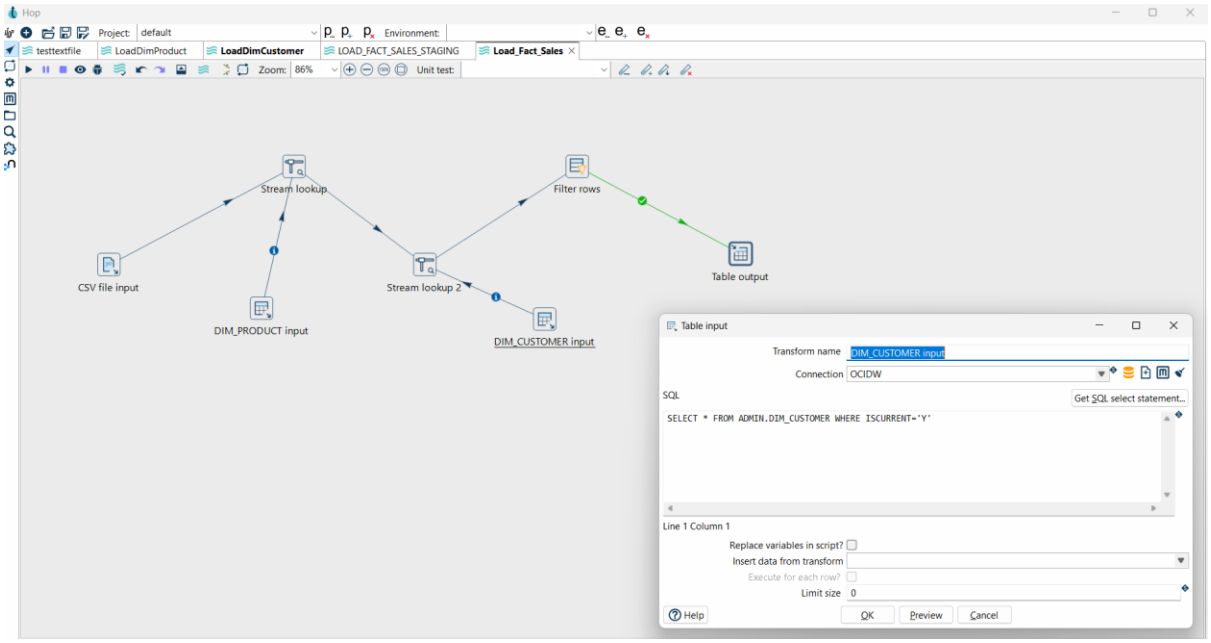


8) DIM_CUSTOMER screen snip after ETL Pipeline execution

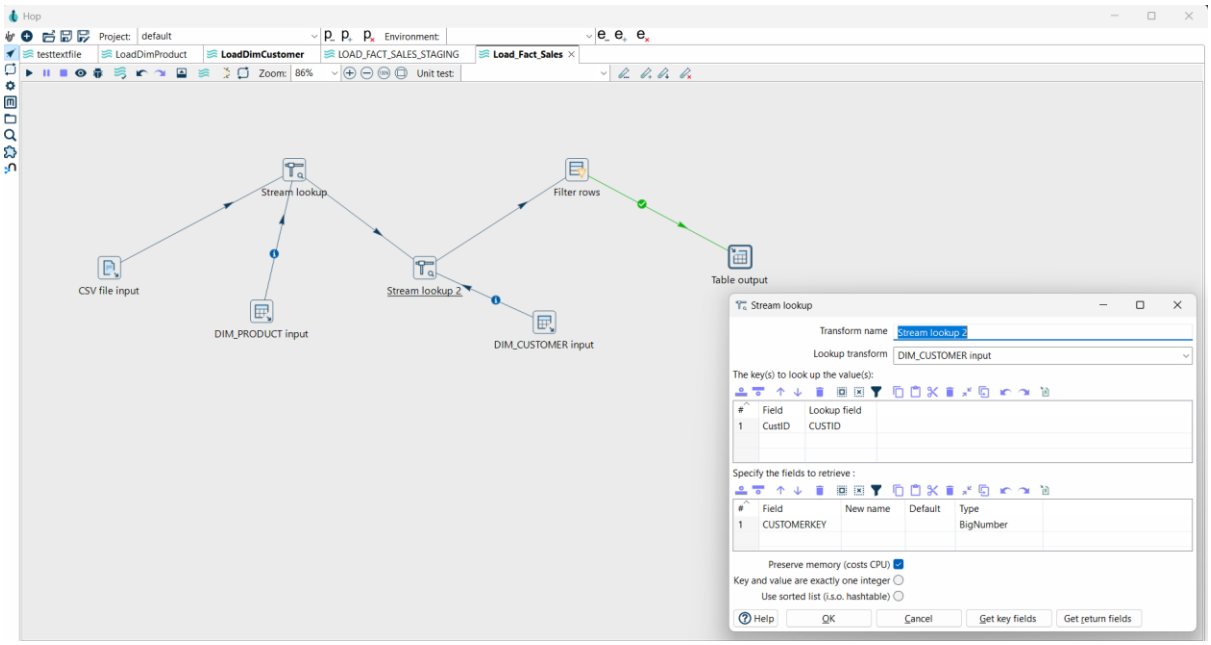


Loading Fact Table

1) *DIM_CUSTOMER input node screen snip*



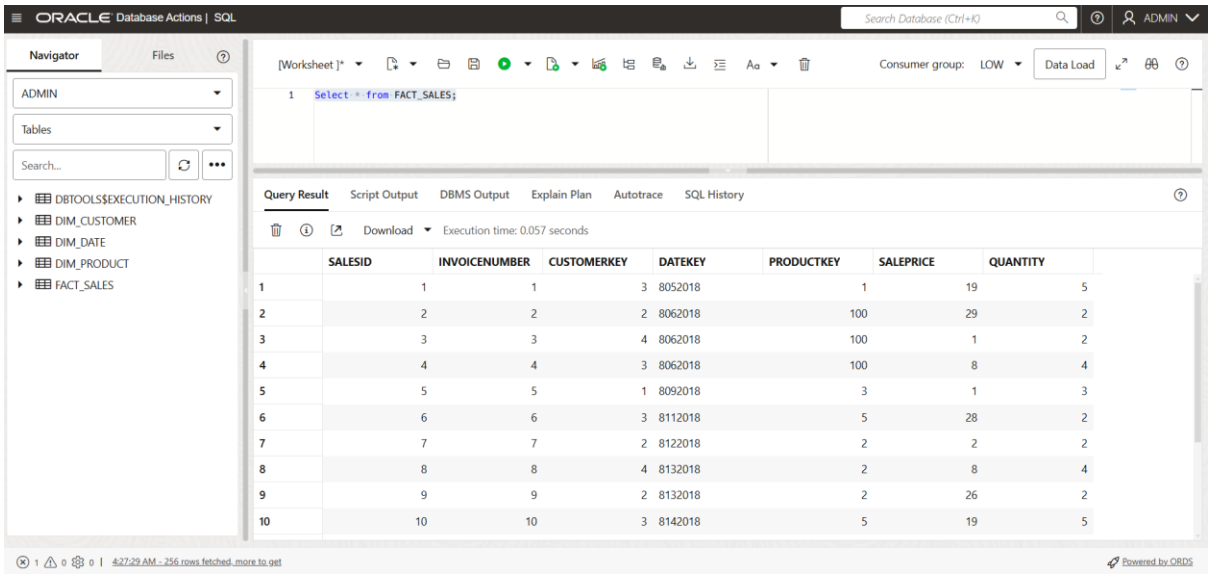
2) *Second stream input node*



3) *Why is no lookup performed in the Date Dimension?*

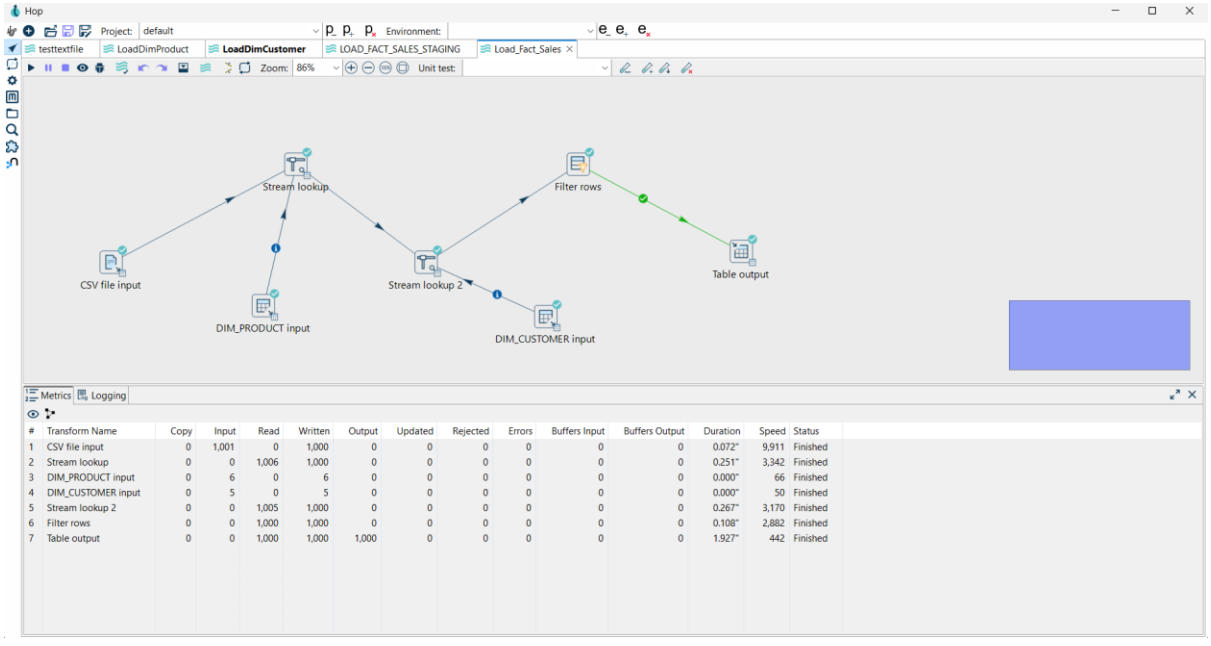
Ans - A lookup is used to identify the current valid record for a business key in the dimension table. However, this is unnecessary for the date dimension because each date has a unique record. Referential integrity is ensured as long as the DATEKEY in both the Date dimension and Fact table share the same format or are transformed to match. This alignment can also be managed by the customer-facing Business Intelligence platform.

4) *FACT_SALES screen snip*



	SALESID	INVOICENUMBER	CUSTOMERKEY	DATEKEY	PRODUCTKEY	SALEPRICE	QUANTITY
1	1	1	3	8052018	1	19	5
2	2	2	2	8062018	100	29	2
3	3	3	4	8062018	100	1	2
4	4	4	3	8062018	100	8	4
5	5	5	1	8092018	3	1	3
6	6	6	3	8112018	5	28	2
7	7	7	2	8122018	2	2	2
8	8	8	4	8132018	2	8	4
9	9	9	2	8132018	2	26	2
10	10	10	3	8142018	5	19	5

5) *Completed Pipeline screen snip*



#	Transform Name	Copy	Input	Read	Written	Output	Updated	Rejected	Errors	Buffers Input	Buffers Output	Duration	Speed	Status
1	CSV file input	0	1,001	0	1,000	0	0	0	0	0	0	0.072"	9,911	Finished
2	Stream lookup	0	0	1,006	1,000	0	0	0	0	0	0	0.251"	3,342	Finished
3	DIM_PRODUCT input	0	6	0	6	0	0	0	0	0	0	0.000"	66	Finished
4	DIM_CUSTOMER input	0	5	0	5	0	0	0	0	0	0	0.000"	50	Finished
5	Stream lookup 2	0	0	1,005	1,000	0	0	0	0	0	0	0.267"	3,170	Finished
6	Filter rows	0	0	1,000	1,000	0	0	0	0	0	0	0.108"	2,882	Finished
7	Table output	0	0	1,000	1,000	1,000	0	0	0	0	0	1.927"	442	Finished

References

1. ChatGPT for paraphrasing the executive summary.