Setting Up Your Cloud Environment & Creating a Warehouse

Summary:

This assignment involved setting up a cloud-based data warehousing environment in Oracle, which was my first experience working with Oracle's cloud platform. The setup process was straightforward, largely due to the simplified environment provided by our professor, where we didn't have to manage cost considerations—a major difference from real-world cloud environments.

Having prior experience working with Azure and Hive databases as both a data analyst and a data engineer, I found that transitioning to Oracle's environment was not entirely new. While there were some differences, the skills I had acquired previously helped me quickly adapt to the Oracle platform.

When it came to building the data warehouse, I opted to use **Structured Query Language (SQL)** over a **Graphical User Interface (GUI)**. Given my three years of experience with SQL, this felt like a more natural and efficient approach for me. I manually created all the dimension and fact tables using SQL queries, defining relationships using primary and foreign keys to ensure data integrity across the warehouse.

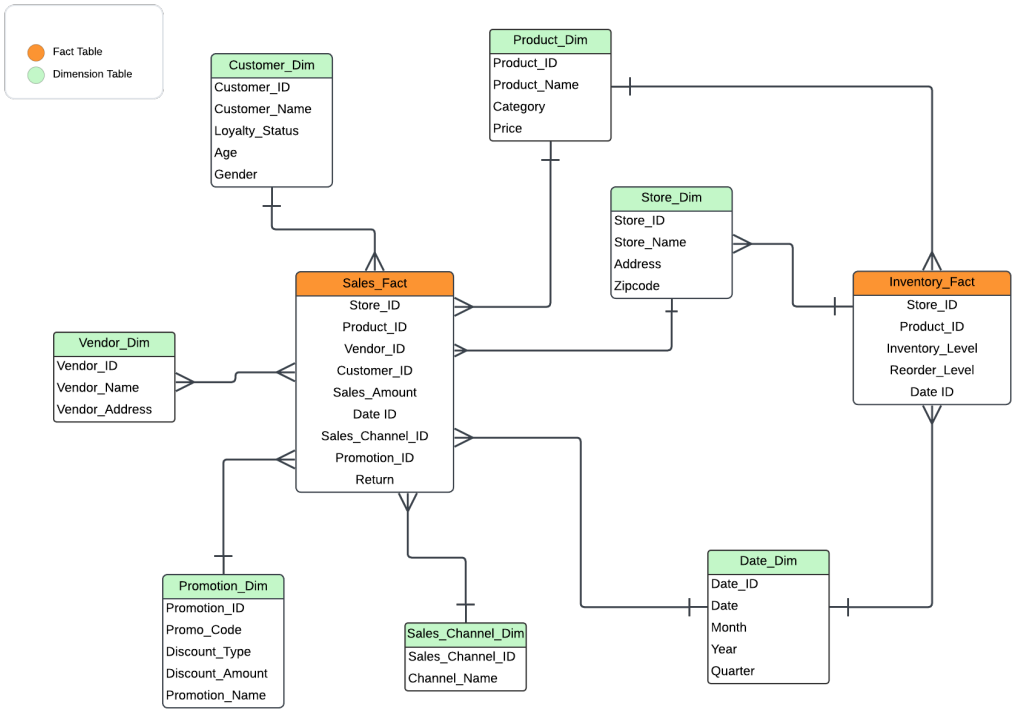
During this process, I encountered two important revisions that I made based on insights from our lectures and practical experience:

1. **Date Dimension Enhancement**: I revised my original approach to the Date Dimension. In line with our coursework, I implemented a structure where the date, time, and related

components (day, month, year, hour, minute, second) are pulled dynamically from a pre-defined script. This ensures that time is managed consistently and efficiently, even when dealing with minute or second-level granularity.

2. **Fact Table Relationships**: Initially, I attempted to connect two fact tables directly. However, I soon realized this approach was leading to inconsistencies in the data and overall complexity in the design. After further research and reflection on best practices, I understood that linking fact tables directly is generally discouraged due to potential performance issues and granularity mismatches. As a result, I disconnected the fact tables and ensured they were linked only through shared dimension tables, following a more stable and scalable design pattern.

Dimensional Model:



Creating All the Dimension Table:

```
-----------------------CREATING ALL THE DIMENSION TABLE----------------------

--CUSTOMER DIM TABLE
CREATE TABLE CUSTOMER_DIM (
    CUSTOMER_ID    NUMBER PRIMARY KEY,
    CUSTOMER_NAME  VARCHAR2(100),
    LOYALTY_STATUS VARCHAR2(50),
    AGE            NUMBER,
    GENDER         VARCHAR2(10)
);

--PRODUCT DIM
CREATE TABLE PRODUCT_DIM (
    PRODUCT_ID   NUMBER PRIMARY KEY,
    PRODUCT_NAME VARCHAR2(100),
    CATEGORY     VARCHAR2(50),
    PRICE        NUMBER(10, 2)
);

--STORE DIM
CREATE TABLE STORE_DIM (
    STORE_ID   NUMBER PRIMARY KEY,
    STORE_NAME VARCHAR2(100),
    ADDRESS    VARCHAR2(200),
    ZIPCODE    VARCHAR2(10)
);

--VENDOR DIM
CREATE TABLE VENDOR_DIM (
    VENDOR_ID      NUMBER PRIMARY KEY,
    VENDOR_NAME    VARCHAR2(100),
    VENDOR_ADDRESS VARCHAR2(200)
);

```

```
--PROMOTION DIM
CREATE TABLE PROMOTION_DIM (
    PROMOTION_ID    NUMBER PRIMARY KEY,
    PROMO_CODE      VARCHAR2(50),
    DISCOUNT_TYPE   VARCHAR2(50),
    DISCOUNT_AMOUNT NUMBER(10, 2),
    PROMOTION_NAME  VARCHAR2(100)
);

--SALES CHANNEL DIM
CREATE TABLE SALES_CHANNEL_DIM (
    SALES_CHANNEL_ID NUMBER PRIMARY KEY,
    CHANNEL_NAME     VARCHAR2(50)
);

--DATE DIM
CREATE TABLE DATE_DIM (
    DATE_ID NUMBER,
    DAY     NUMBER(2),
    MONTH   NUMBER(2),
    YEAR    NUMBER(4),
    HOUR    NUMBER(2),
    MINUTE  NUMBER(2),
    SECOND  NUMBER(2),
    PRIMARY KEY ( DATE_ID )
);
```

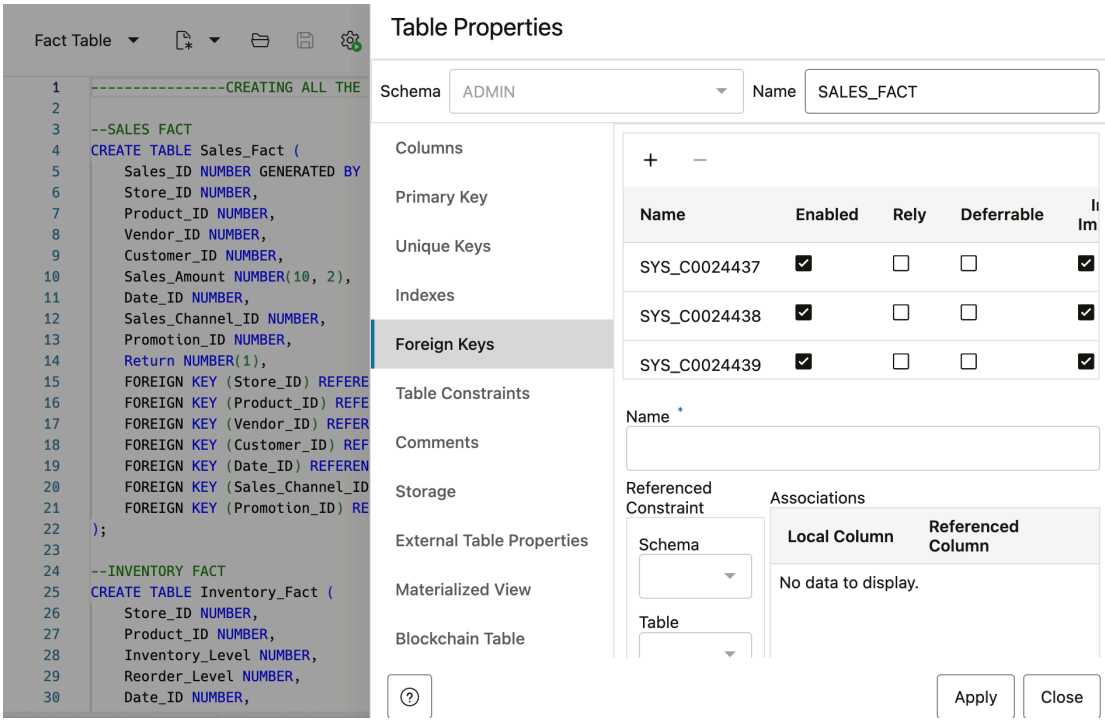
Creating All the Fact Table:

```
----------------CREATING ALL THE FACT TABLE----------------------

--SALES FACT
CREATE TABLE Sales_Fact (
    Sales_ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Store_ID NUMBER,
    Product_ID NUMBER,
    Vendor_ID NUMBER,
    Customer_ID NUMBER,
    Sales_Amount NUMBER(10, 2),
    Date_ID NUMBER,
    Sales_Channel_ID NUMBER,
    Promotion_ID NUMBER,
    Return NUMBER(1),
    FOREIGN KEY (Store_ID) REFERENCES Store_Dim(Store_ID),
    FOREIGN KEY (Product_ID) REFERENCES Product_Dim(Product_ID),
    FOREIGN KEY (Vendor_ID) REFERENCES Vendor_Dim(Vendor_ID),
    FOREIGN KEY (Customer_ID) REFERENCES Customer_Dim(Customer_ID),
    FOREIGN KEY (Date_ID) REFERENCES Date_Dim(Date_ID),
    FOREIGN KEY (Sales_Channel_ID) REFERENCES Sales_Channel_Dim(Sales_Channel_ID),
    FOREIGN KEY (Promotion_ID) REFERENCES Promotion_Dim(Promotion_ID)
);

--INVENTORY FACT
CREATE TABLE Inventory_Fact (
    Store_ID NUMBER,
    Product_ID NUMBER,
    Inventory_Level NUMBER,
    Reorder_Level NUMBER,
    Date_ID NUMBER,
    PRIMARY KEY (Store_ID, Product_ID, Date_ID),
    FOREIGN KEY (Store_ID) REFERENCES Store_Dim(Store_ID),
    FOREIGN KEY (Product_ID) REFERENCES Product_Dim(Product_ID),
    FOREIGN KEY (Date_ID) REFERENCES Date_Dim(Date_ID)
);

```

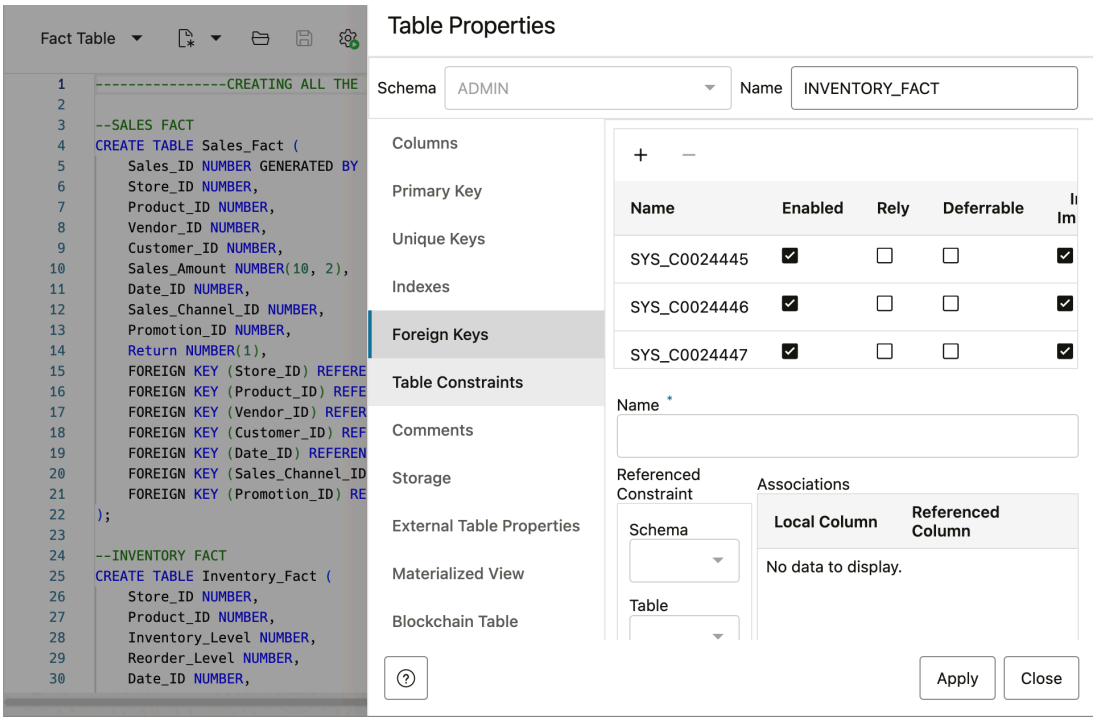
Note: I have created all the tables using SQL commands so the DDL statement attachment will be same as my SQL code.

Foreign Key Relation:

Sales Fact:

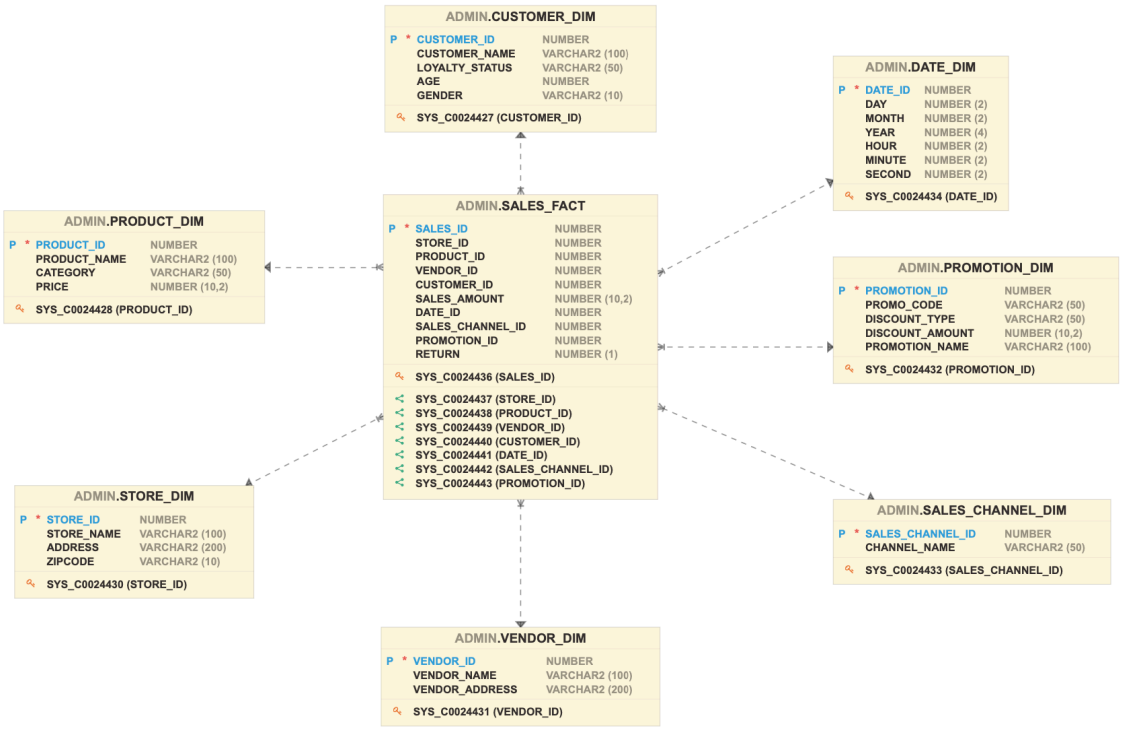


Inventory Fact:

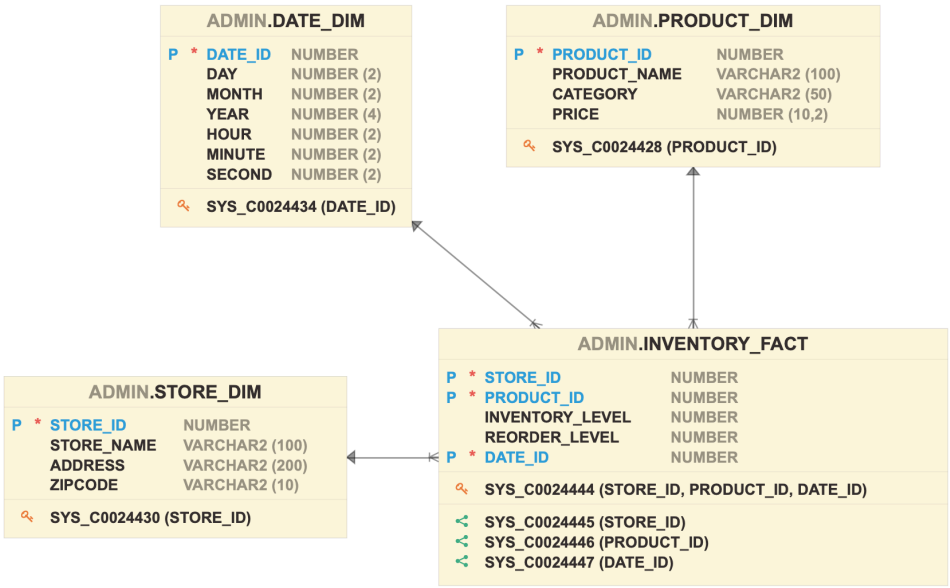


Data Warehouse Diagram:

Sales Fact:



Inventory Fact:



Creating a ETL pipeline

Summary:

It was my first-time using Apache Hop, although I have previous experience with ETL tools like Informatica and Talend. The primary objective was to populate a data warehouse hosted on Oracle Cloud, integrating data from various sources into dimension and fact tables. My experience as a data analyst allowed me to understand the technical requirements and navigate challenges efficiently, especially in handling data transformations and loading procedures.

Key Observations

1. **"Date" Column in Date_dim Table**: One critical observation was the usage of "Date" as a column name in the Date_dim table. "Date" is a reserved keyword in Oracle, which could cause errors during data migrations or when writing ETL pipeline scripts. This was an important lesson, as I had encountered similar issues in past projects and was able to address it here proactively. This observation was also discussed in class, highlighting the importance of using non-reserved names for column identifiers to avoid potential conflicts in SQL-based environments.
2. **Missing ProductKey Column in Product Table**: Another issue I noticed was the discrepancy between the product table structure and its diagram. The Product table should have 10 columns, but the diagram only depicts 9, with ProductKey missing. This missing column could lead to data integrity issues when performing lookups or joins, especially in the fact table, as ProductKey often serves as a crucial link between dimension and fact tables. From my experience, it’s vital to ensure that diagrams accurately represent table structures to prevent misunderstandings or errors during implementation.

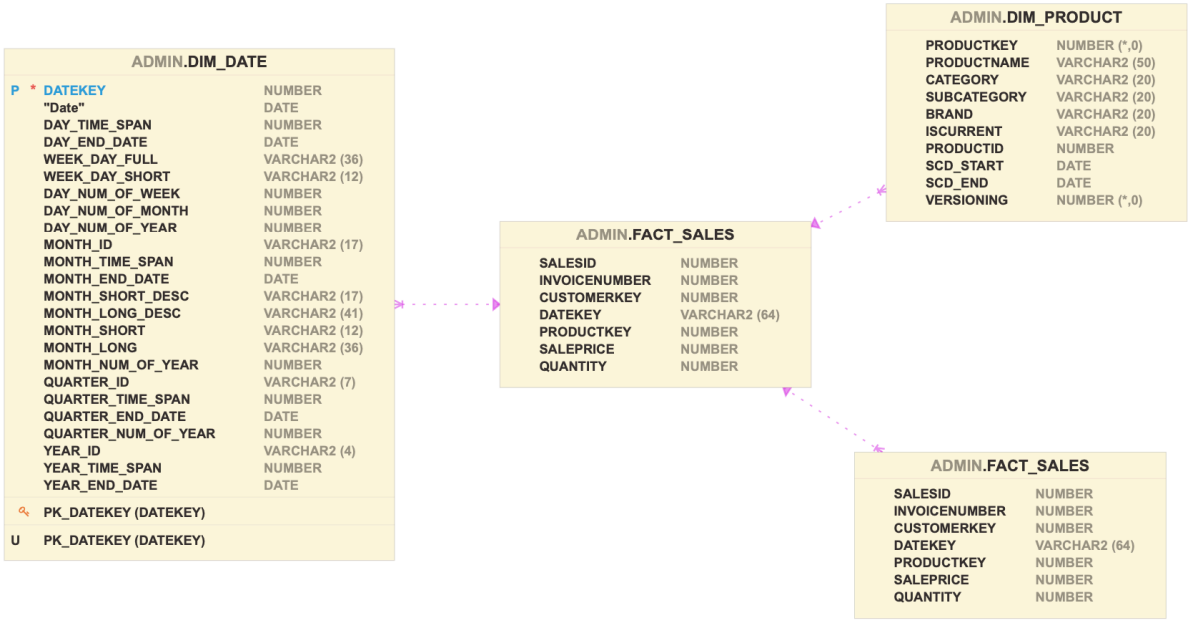
Based on the diagram generated, what is this database missing that you'd expect to see? Why might it be missing this component?

The database is missing **foreign keys**, which are used to establish relationships between tables and ensure data integrity. These are likely missing because the ETL pipeline is handling the relationships in stream lookup by mapping data to the correct columns during the data loading process. While this approach works, it relies on the ETL process being accurate, as the database itself won't automatically prevent errors like invalid or missing references. Adding foreign keys could help explicitly define relationships and improve data integrity.

You might have noticed we're not doing a lookup for the date dimension. Write 1-2 sentences detailing why we don't need to. You'll be able to figure this out likely by looking at the data in the fact CSV and the date dimension.

The lookup for the date dimension is not required because the sales update table already contains a date field that directly corresponds to the fact table. This is handled in the ETL pipeline by extracting the date field and using it as-is, eliminating the need for a separate lookup process.

Generate a diagram of the database you imported and attach a screenshot to your report.



Open a new query and type “SELECT * FROM Dim_Date” and click “Execute”. Paste a screenshot of the result in your report (with a sample of rows).

```
49
50  select * from DIM_DATE;
51
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.025 seconds

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DA
1	20180101	1/1/2018, 12:00:00 A	1	1/1/2018, 12:00:00 A	Monday	MON	
2	20180102	1/2/2018, 12:00:00 A	1	1/2/2018, 12:00:00 A	Tuesday	TUE	
3	20180103	1/3/2018, 12:00:00 A	1	1/3/2018, 12:00:00 A	Wednesday	WED	

```
50  select min("Date"),max("Date") from DIM_DATE;
51
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.051 seconds

	MIN("DATE")	MAX("DATE")
1	1/1/2018, 12:00:00 A	3/19/2026, 12:00:00

Run the “SELECT * FROM Dim_Date” query again and paste a screenshot in your report, along with a copy of the full updated script you ran.

--after changing the date

```
49  select * from DIM_DATE;
50  --select min("Date"),max("Date") from DIM_DATE;
51
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.024 seconds

	DATEKEY	DATE	DAY_TIME_SPAN	DAY_END_DATE	WEEK_DAY_FULL	WEEK_DAY_SHOR	DA
1	20160101	1/1/2016, 12:00:00 A	1	1/1/2016, 12:00:00 A	Friday	FRI	
2	20160102	1/2/2016, 12:00:00 A	1	1/2/2016, 12:00:00 A	Saturday	SAT	
3	20160103	1/3/2016, 12:00:00 A	1	1/3/2016, 12:00:00 A	Sunday	SUN	
4	20160104	1/4/2016, 12:00:00 A	1	1/4/2016, 12:00:00 A	Monday	MON	

```
50  select min("Date"),max("Date") from DIM_DATE;
51
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

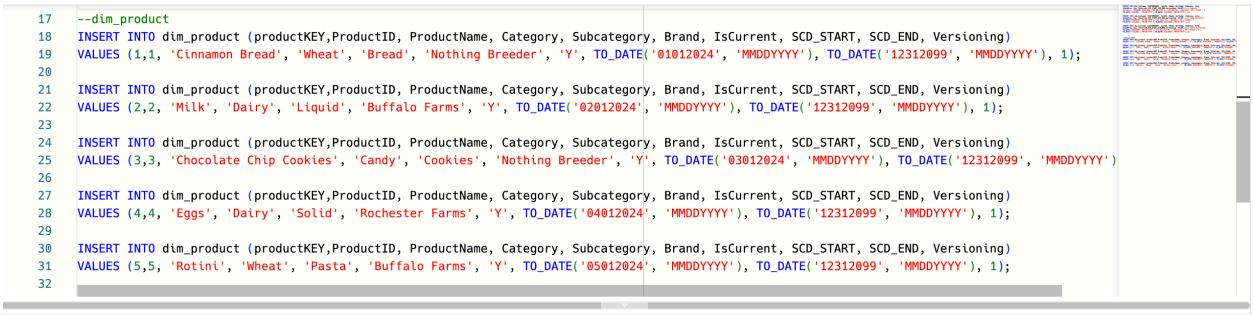
Download | Execution time: 0.056 seconds

	MIN("DATE")	MAX("DATE")
1	1/1/2016, 12:00:00 A	3/18/2024, 12:00:00

```
--script

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,
--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,
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,
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 <= 3000)
ORDER BY CurrDate
;
ALTER TABLE DIM_DATE
ADD CONSTRAINT pk_datekey PRIMARY KEY (DATEKEY);
select * from DIM_DATE;
select min("Date"),max("Date") from DIM_DATE;
```

Once complete, run a “Select * from Dim_Product” query, take a screenshot, and paste the result in your report.



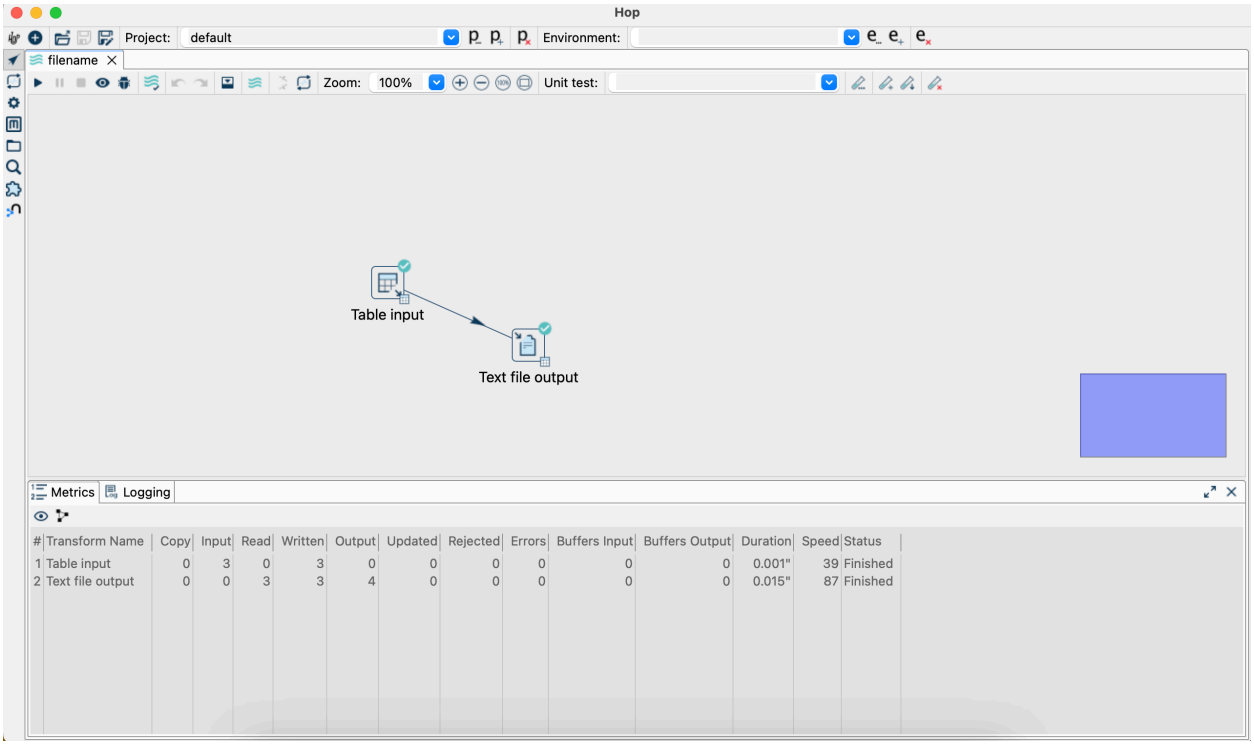
```
--dim_product
INSERT INTO dim_product (productKEY,ProductID, ProductName, Category, Subcategory, Brand, IsCurrent, SCD_START, SCD_END, Versioning)
VALUES (1,1, 'Cinnamon Bread', 'Wheat', 'Bread', 'Nothing Breeder', 'Y', TO_DATE('01012024', 'MMDDYYYY'), TO_DATE('12312099', 'MMDDYYYY'), 1);

INSERT INTO dim_product (productKEY,ProductID, ProductName, Category, Subcategory, Brand, IsCurrent, SCD_START, SCD_END, Versioning)
VALUES (2,2, 'Milk', 'Dairy', 'Liquid', 'Buffalo Farms', 'Y', TO_DATE('02012024', 'MMDDYYYY'), TO_DATE('12312099', 'MMDDYYYY'), 1);

INSERT INTO dim_product (productKEY,ProductID, ProductName, Category, Subcategory, Brand, IsCurrent, SCD_START, SCD_END, Versioning)
VALUES (3,3, 'Chocolate Chip Cookies', 'Candy', 'Cookies', 'Nothing Breeder', 'Y', TO_DATE('03012024', 'MMDDYYYY'), TO_DATE('12312099', 'MMDDYYYY')

INSERT INTO dim_product (productKEY,ProductID, ProductName, Category, Subcategory, Brand, IsCurrent, SCD_START, SCD_END, Versioning)
VALUES (4,4, 'Eggs', 'Dairy', 'Solid', 'Rochester Farms', 'Y', TO_DATE('04012024', 'MMDDYYYY'), TO_DATE('12312099', 'MMDDYYYY'), 1);

INSERT INTO dim_product (productKEY,ProductID, ProductName, Category, Subcategory, Brand, IsCurrent, SCD_START, SCD_END, Versioning)
VALUES (5,5, 'Rotini', 'Wheat', 'Pasta', 'Buffalo Farms', 'Y', TO_DATE('05012024', 'MMDDYYYY'), TO_DATE('12312099', 'MMDDYYYY'), 1);
```

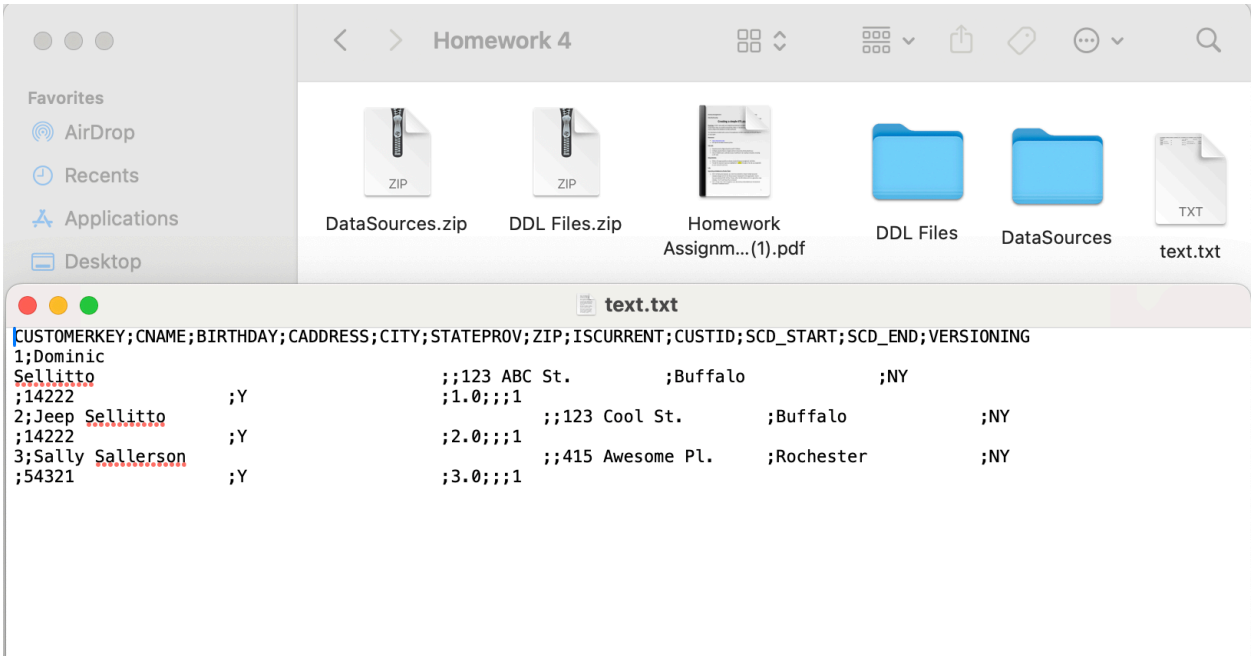
Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.006 seconds

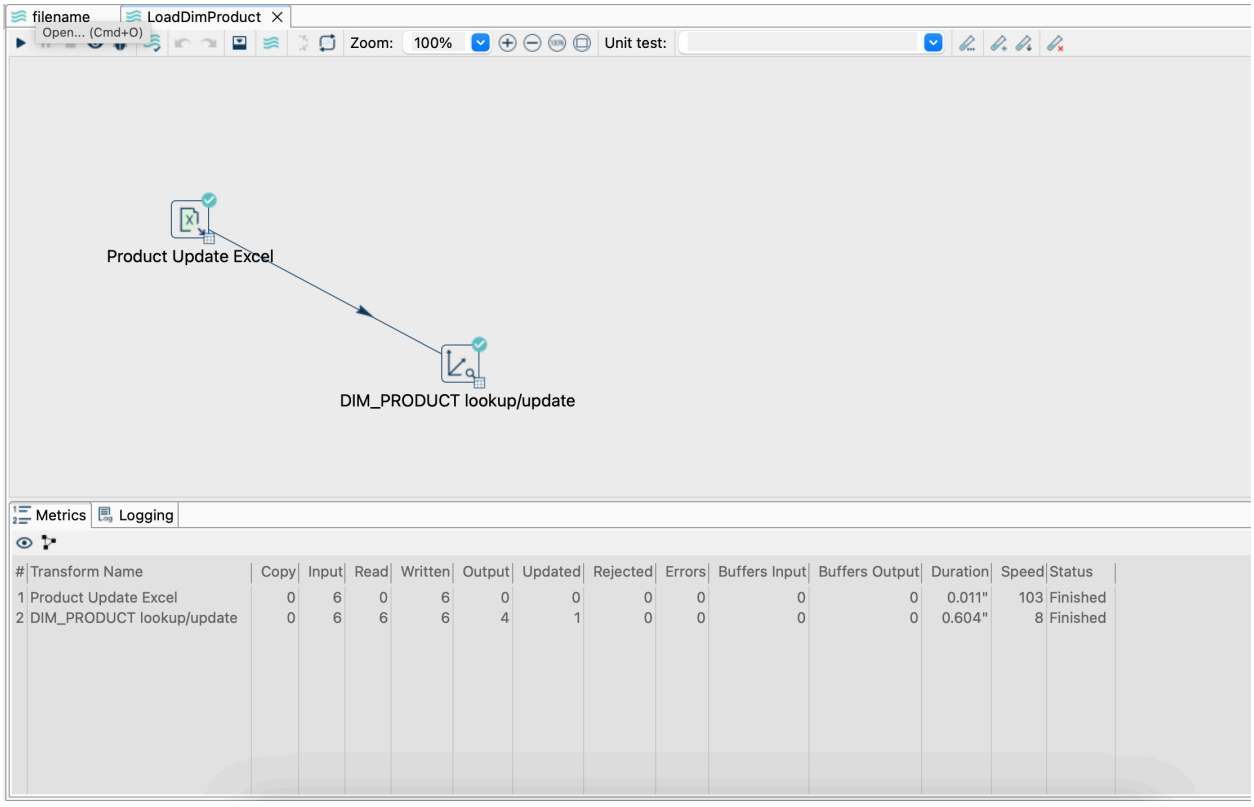
	PRODUCTNAME	CATEGORY	SUBCATEGORY	BRAND	ISCURRENT	PRODUCTID	SCD_START	SCD_END	VERSION
1	Cinnamon Bread	Wheat	Bread	Nothing Breeder	Y	1	1/1/2024, 12:00:00 A	12/31/2099, 12:00:0(	
2	Milk	Dairy	Liquid	Buffalo Farms	Y	2	2/1/2024, 12:00:00 A	12/31/2099, 12:00:0(	
3	Chocolate Chip Cool	Candy	Cookies	Nothing Breeder	Y	3	3/1/2024, 12:00:00 A	12/31/2099, 12:00:0(	
4	Eggs	Dairy	Solid	Rochester Farms	Y	4	4/1/2024, 12:00:00 A	12/31/2099, 12:00:0(	
5	Rotini	Wheat	Pasta	Buffalo Farms	Y	5	5/1/2024, 12:00:00 A	12/31/2099, 12:00:0(	

Add this text file to your final submission for the assignment and add a screenshot of your pipeline to your report.





Back in Oracle Cloud, run a “SELECT * FROM DIM_PRODUCT” command and look at the results. This is a small enough dataset that you can check to see if things worked as you expected them to. If they did, grab a screenshot of the results. Include this screenshot in your report.



#	Transform Name	Copy	Input	Read	Written	Output	Updated	Rejected	Errors	Buffers Input	Buffers Output	Duration	Speed	Status
1	Product Update Excel	0	6	0	6	0	0	0	0	0	0	0.011"	103	Finished
2	DIM_PRODUCT lookup/update	0	6	6	6	4	1	0	0	0	0	0.604"	8	Finished

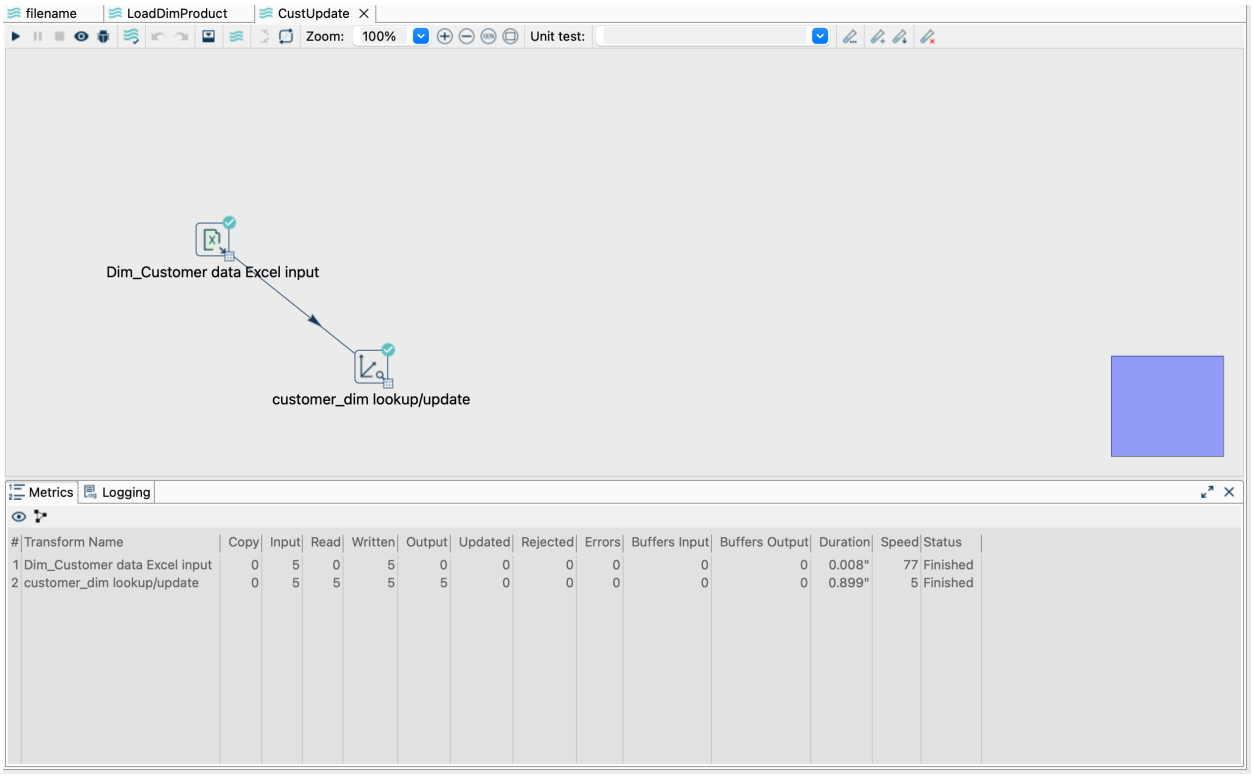
```
select * from dim_product
```

sult | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.003 seconds

PRODUCTKEY	PRODUCTNAME	CATEGORY	SUBCATEGORY	BRAND	ISCURRENT	PRODUCTID	SCD_START	SCD_END
0	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*	*(null)*
100	Cinnamon Bread Loa	Wheat	Bread	Nothing Breader	Y	1	11/14/2024, 7:30:34	12/31/219
101	Chocolate Chip Cool	Candy	Cookies	Nothing Breader	Y	3	11/14/2024, 7:30:34	12/31/219
102	Eggs	Poultry	Solid	Rochester Farms	Y	4	11/14/2024, 7:30:34	12/31/219
103	Sugary Cereal	Wheat	Cereal	Food For You	Y	6	11/14/2024, 7:30:34	12/31/219
1	Cinnamon Bread Loa	Wheat	Bread	Nothing Breeder	N	1	1/1/2024, 12:00:00 A	11/14/202
2	Milk	Dairy	Liquid	Buffalo Farms	Y	2	2/1/2024, 12:00:00 A	12/31/209
3	Chocolate Chip Cool	Candy	Cookies	Nothing Breeder	N	3	3/1/2024, 12:00:00 A	11/14/202
4	Eggs	Dairy	Solid	Rochester Farms	N	4	4/1/2024, 12:00:00 A	11/14/202
5	Rotini	Wheat	Pasta	Buffalo Farms	Y	5	5/1/2024, 12:00:00 A	12/31/209

Take a screenshot of your Apahe Hop Pipeline and include it in your report. f. Run a “SELECT * FROM Dim_Customer” query and paste a screenshot of the output in your report.



#	Transform Name	Copy	Input	Read	Written	Output	Updated	Rejected	Errors	Buffers Input	Buffers Output	Duration	Speed	Status
1	Dim_Customer data Excel input	0	5	0	5	0	0	0	0	0	0	0.008"	77	Finished
2	customer_dim lookup/update	0	5	5	5	5	0	0	0	0	0	0.899"	5	Finished

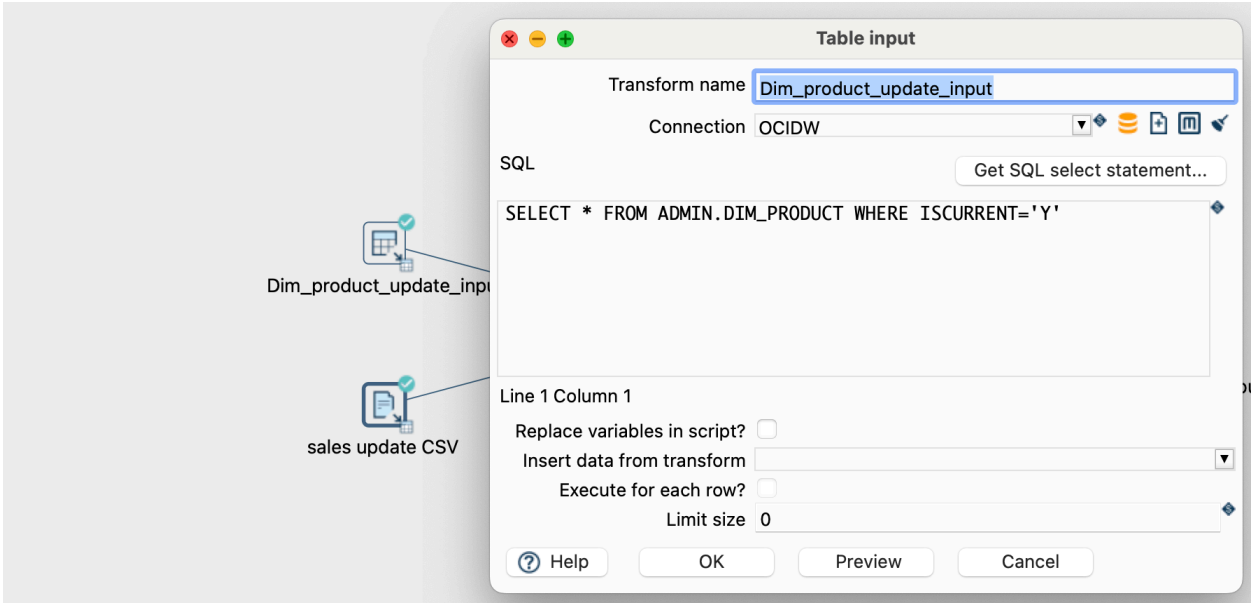
sult Script Output DBMS Output Explain Plan Autotrace SQL History

Download Execution time: 0.003 seconds

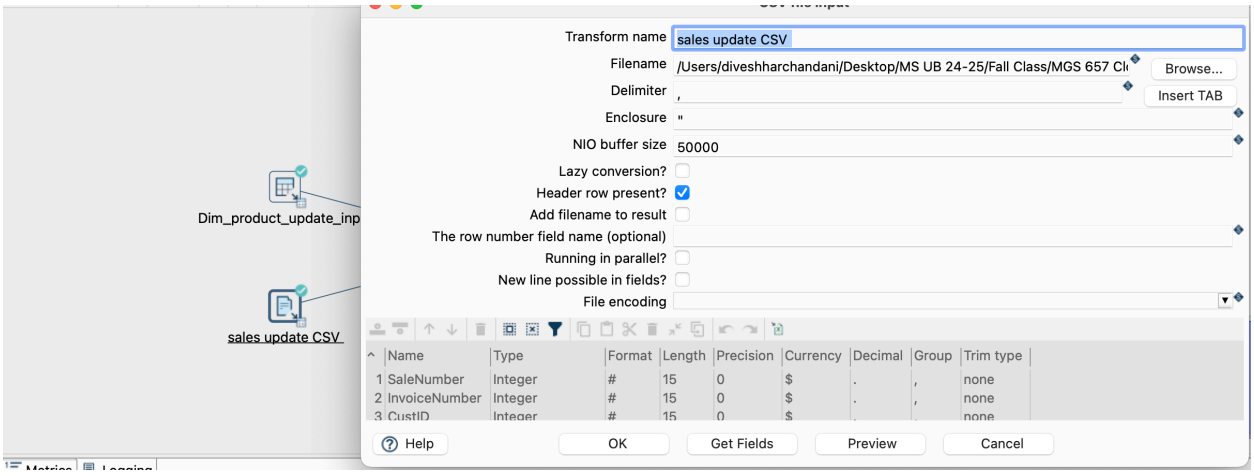
CUSTOMERKEY	CNAME	BIRTHDAY	CADDRESS	CITY	STATEPROV	ZIP	ISCURRENT	CUSTID
1	Dominic Sellitto	1/1/1956, 12:00:00 A	123 ABC St.	Buffalo	NY	14222	N	
2	Jeep Sellitto	2/2/1979, 12:00:00 A	123 Cool St.	Buffalo	NY	14222	N	
3	Sally Sallerson	3/3/1989, 12:00:00 A	415 Awesome Pl.	Rochester	NY	54321	N	
1002	Dominic Sellitto	1/1/1956, 12:00:00 A	123 New St.	Rochester	NY	14321	Y	
1003	Jeep Jeeperson	2/2/1979, 12:00:00 A	123 Cool St.	Buffalo	NY	14043	Y	
1004	Sally Sallerson	3/3/1989, 12:00:00 A	415 Awesome Pl.	Rochester	NY	54321	Y	
1005	James Bond	4/4/1999, 12:00:00 A	543 Bond Rd.	Buffalo	NY	14222	Y	
1006	Jennifer Lopez	5/5/2009, 12:00:00 A	91 Perfect Ave.	Rochester	NY	14321	Y	

LOAD_FACT_SALES_STAGING

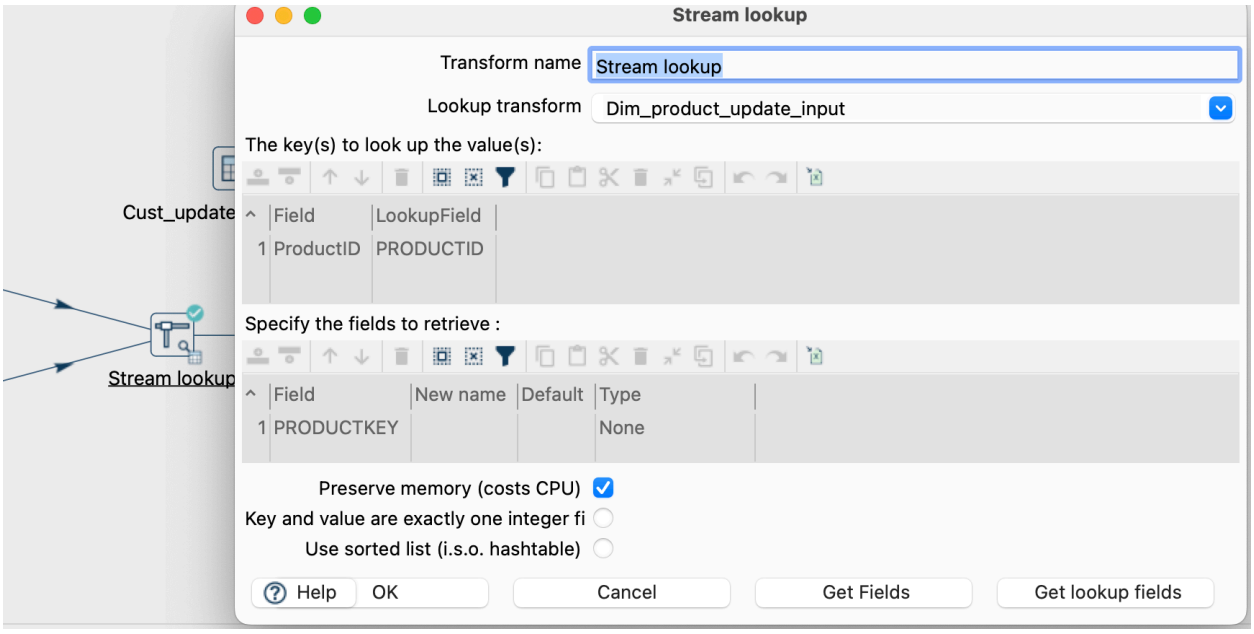
Dim_product_update_input



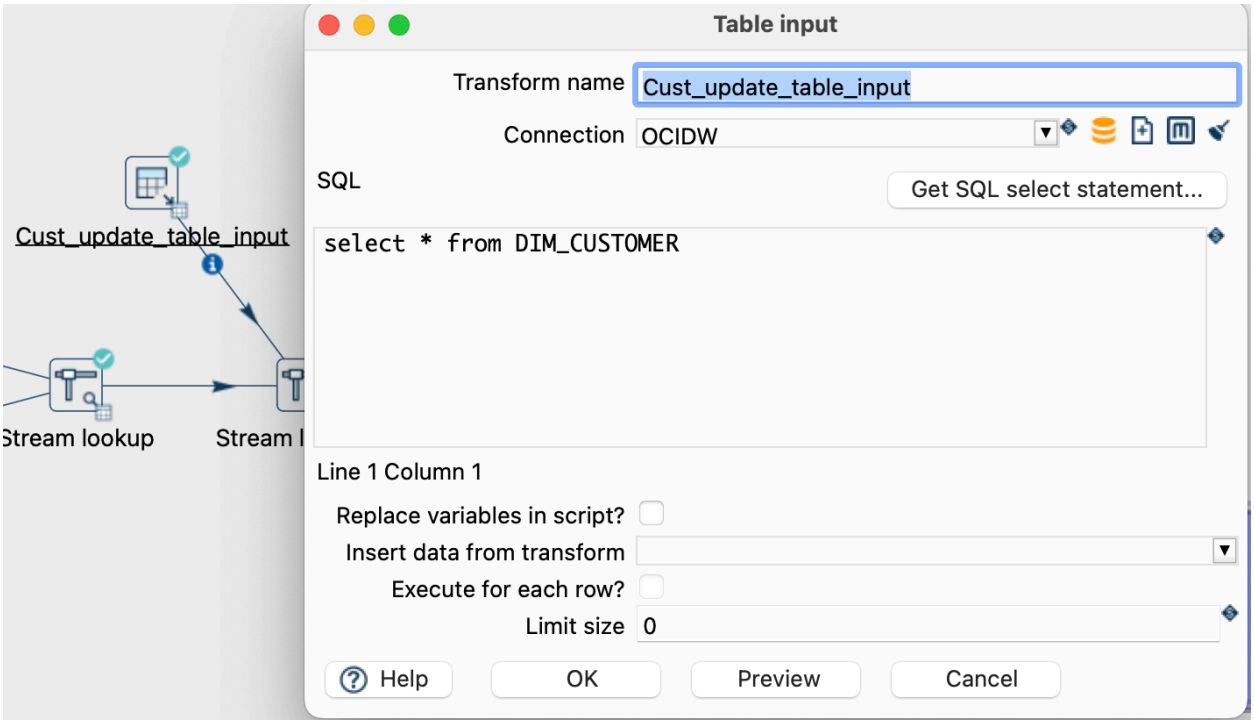
sales update CSV



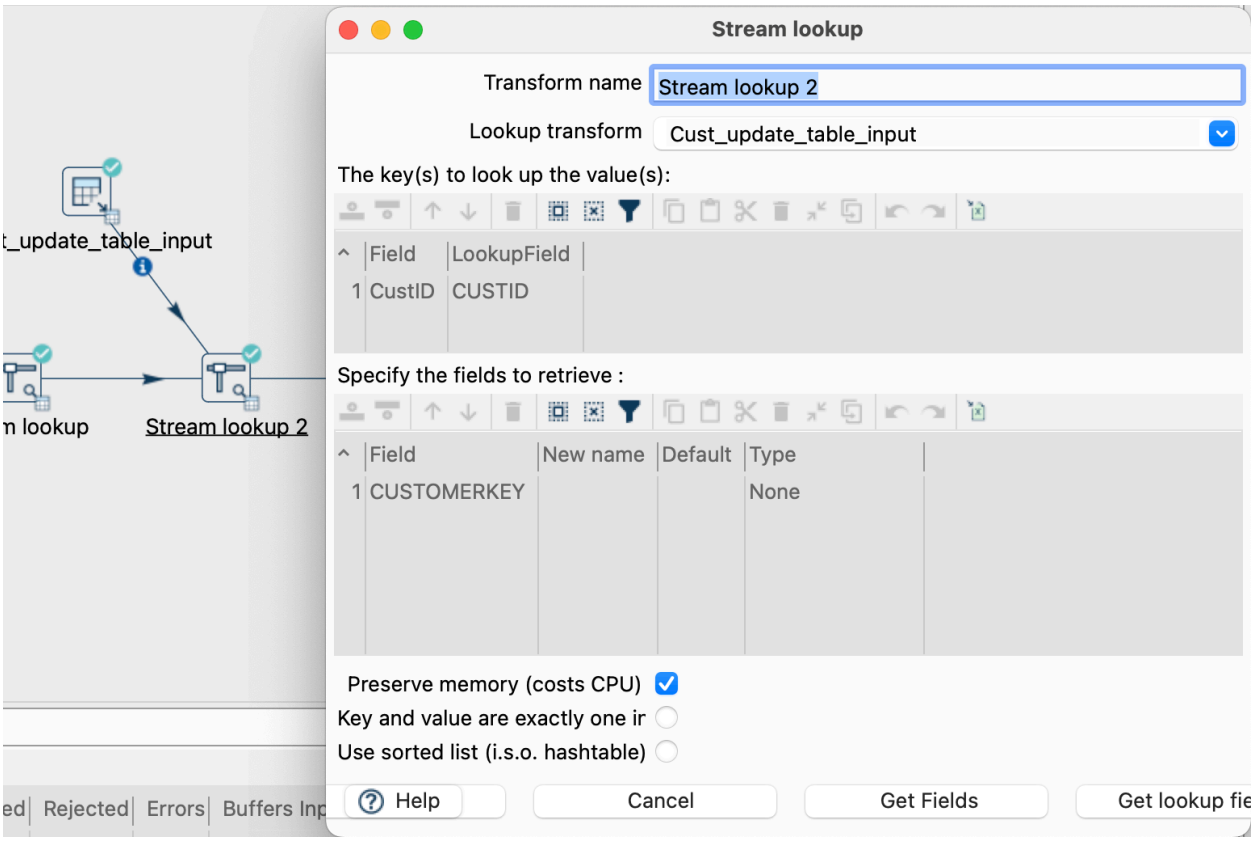
Stream lookup



Cust_update_table_input



Stream lookup 2



Filter rows

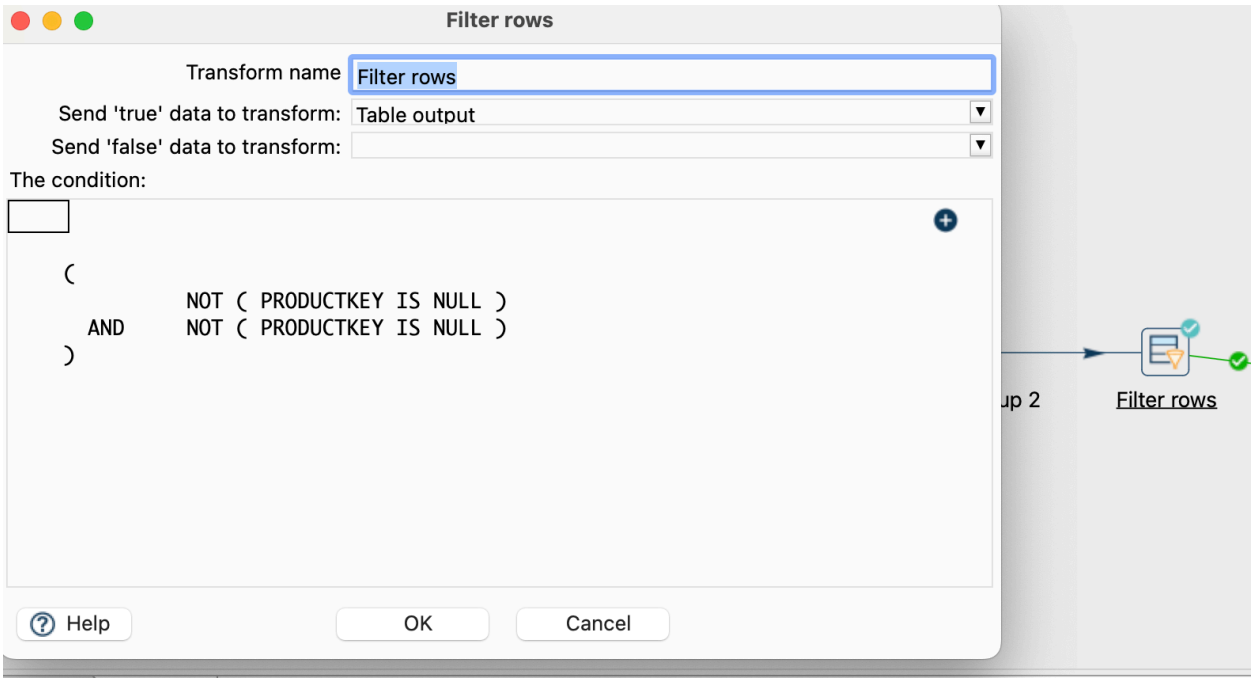
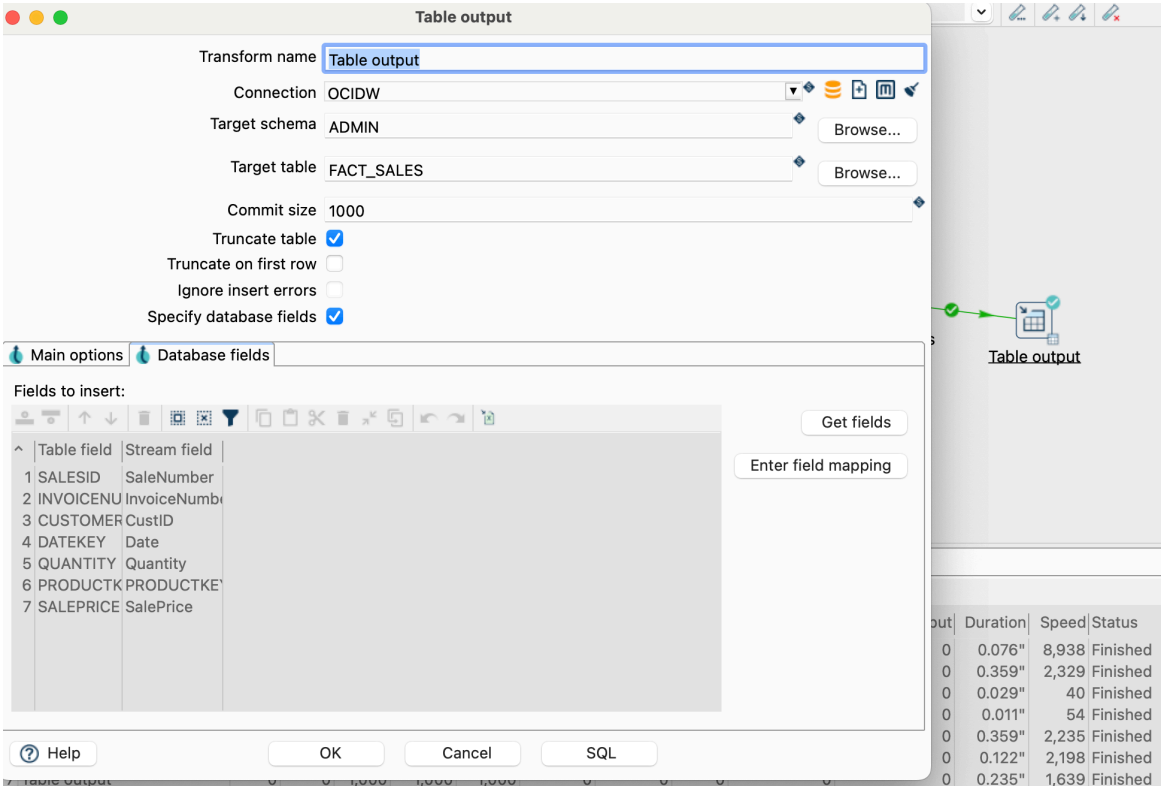
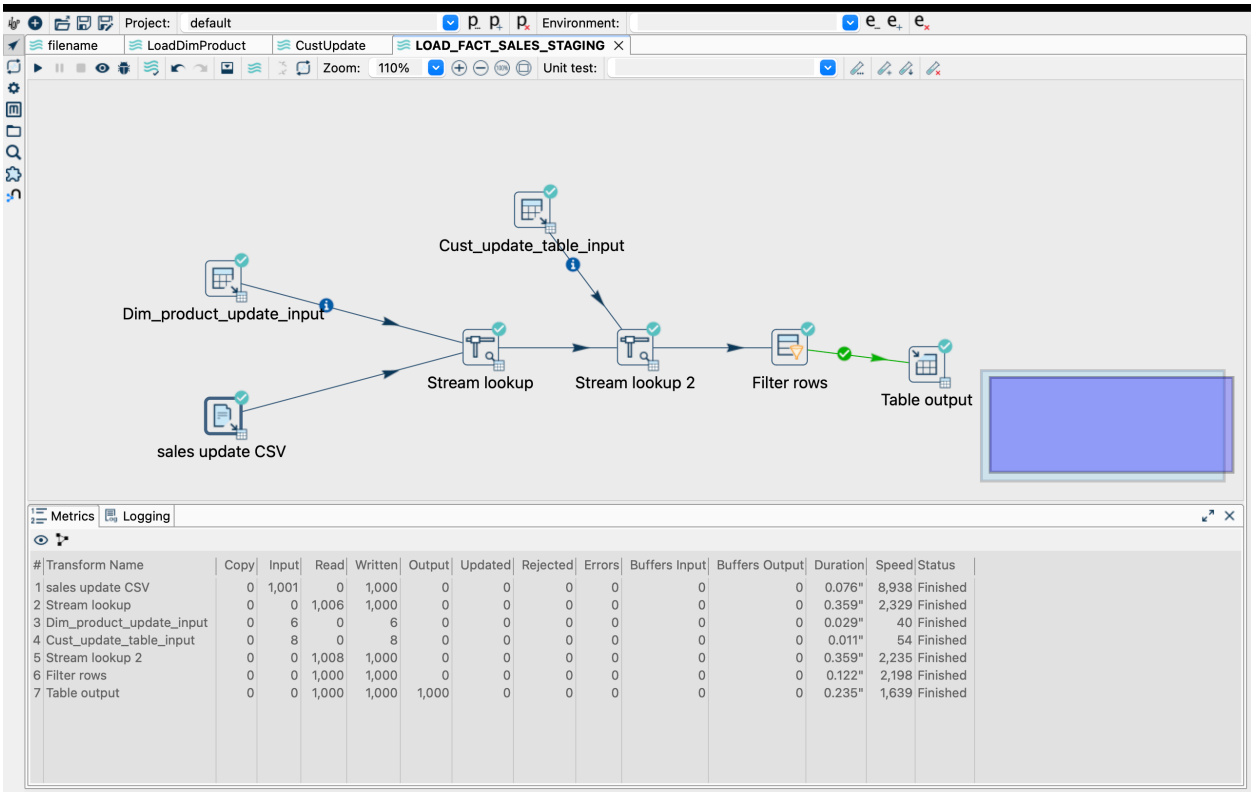


Table output



Entire Flow



Fact_Sales Output

```
4  SELECT * FROM Fact_sales
5
6
```

ery Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download | Execution time: 0.012 seconds

SALESID	INVOICENUMBER	CUSTOMERKEY	DATEKEY	PRODUCTKEY	SALEPRICE	QUANTITY
1	1	3	8052018	100	19	5
2	2	2	8062018	102	29	2
3	3	5	8062018	102	1	2
4	4	3	8062018	102	8	4
5	5	1	8092018	101	1	3
6	6	4	8112018	5	28	2
7	7	2	8122018	2	2	2
8	8	5	8132018	2	8	4
9	9	2	8132018	2	26	2
10	10	3	8142018	5	19	5
11	11	5	8142018	100	28	1
12	12	2	8162018	103	30	1
13	13	4	8212018	5	15	5
14	14	5	8232018	102	18	4

Count from Fact_sales

```
SELECT count(*) FROM Fact_sales
```

Result | Script Output | DBMS Output | Explain Plan

Download | Execution time: 0.019 seconds

COUNT(*)
1000

Summary

This assignment focused on analyzing and visualizing sales data to identify key trends, patterns, and performance metrics. The primary objective was to provide actionable insights into sales performance across different brands, categories, and stores, utilizing Tableau for effective data visualization. By exploring sales data at multiple levels, the assignment aimed to uncover meaningful insights that could support decision-making and drive business growth.

The first step involved cleaning and preparing the dataset to ensure accurate and reliable analysis. This included aggregating sales data by brand, category, and store and calculating total and average sales figures. Key performance indicators (KPIs) such as total sales count, category contributions, and brand-specific performance were identified for visualization.

The dashboard created provides a clear and interactive view of the data. A **Total Sales Overall** KPI highlights the cumulative sales count, giving an immediate sense of overall performance. A **Total Sales by Brand** bar chart compares sales contributions by brand, with an average line to benchmark performance. A **Total Sales by Category** donut chart visually represents the proportion of sales by product categories, making it easy to see which categories are most significant. Lastly, a **Month-over-Month Sales** line chart tracks trends in actual and estimated sales over time, offering insights into seasonality and forecasting accuracy. Filters for city and category allow for dynamic exploration of the data.

This assignment demonstrated the value of visual analytics in presenting complex data in an accessible format. By using Tableau, the assignment achieved a balance between data depth and simplicity, enabling stakeholders to make informed decisions based on clear and interactive visualizations. Overall, the activity highlighted the importance of structured data analysis in identifying trends, monitoring performance, and uncovering opportunities for growth.

Document the field in your report.

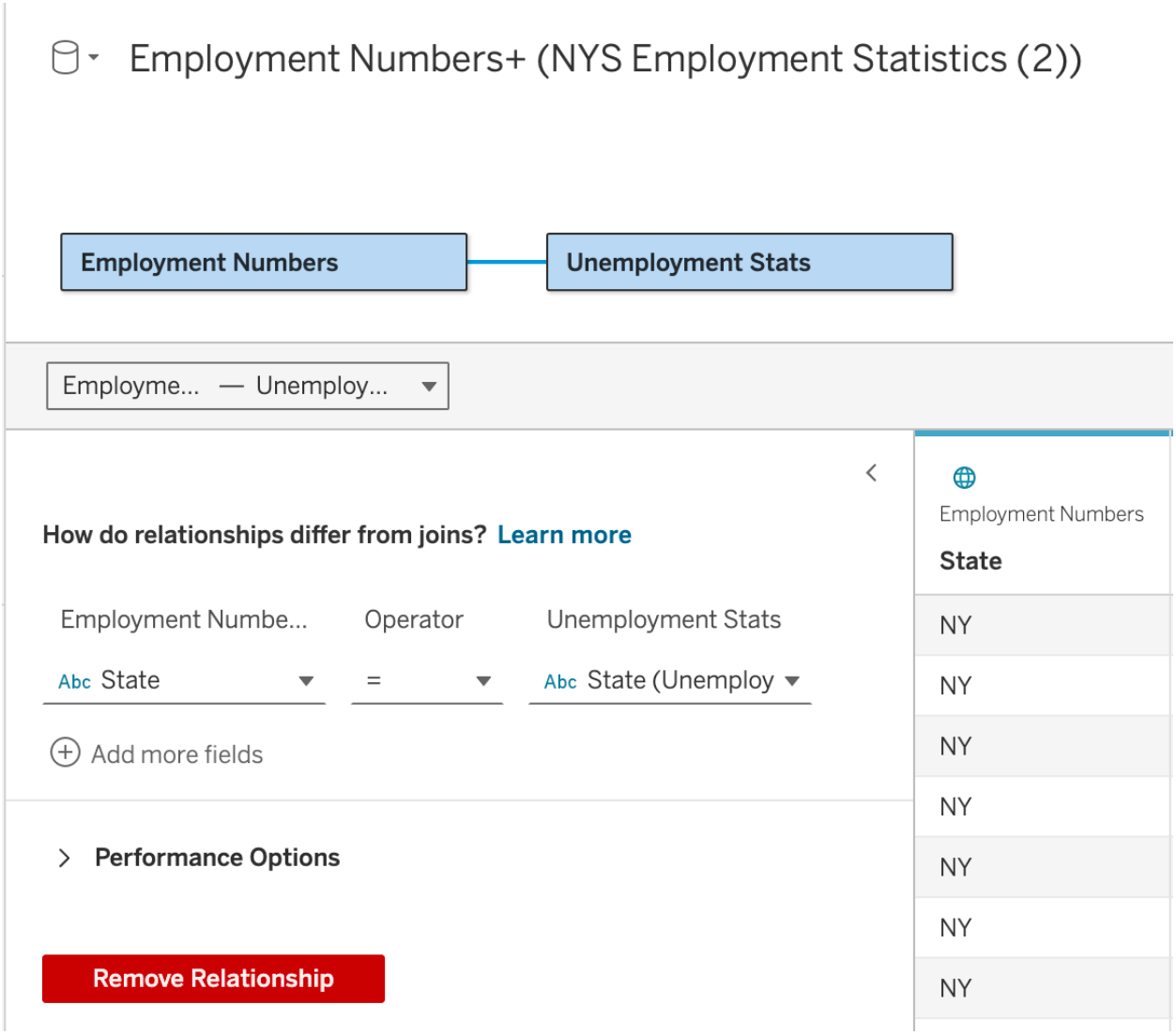
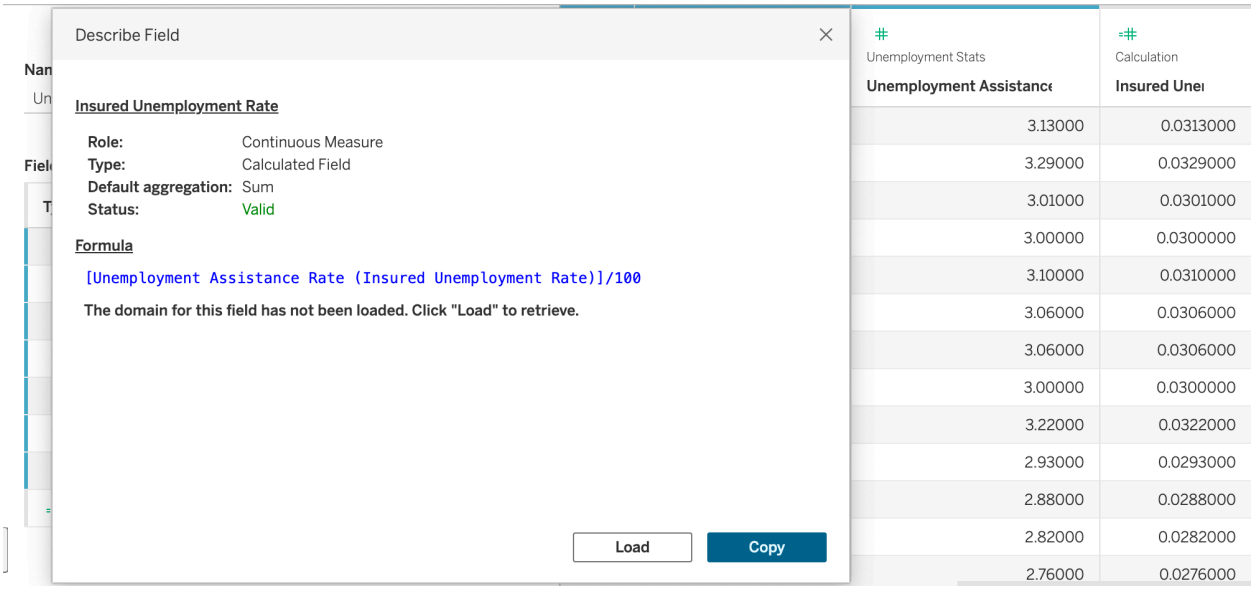


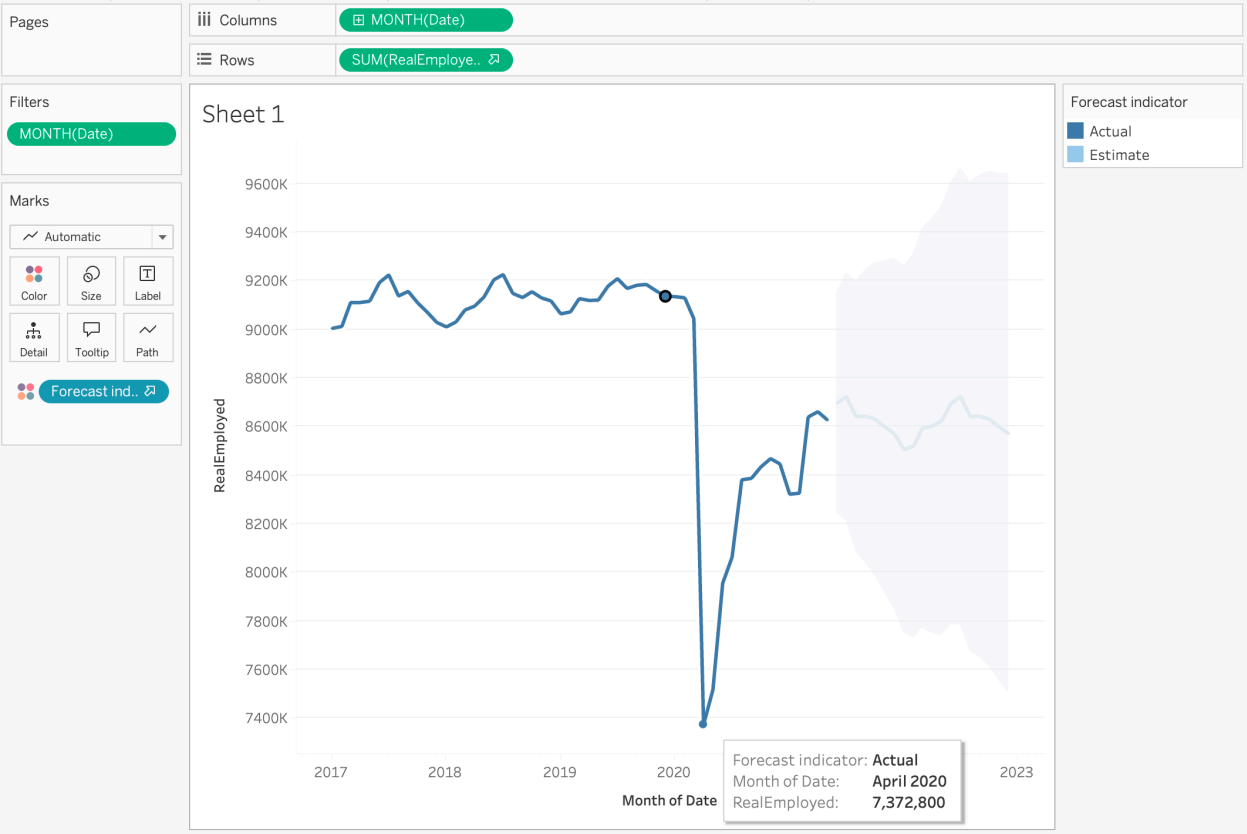
Table Employment Numbers and Unemployment Stats have same column State so that is the relation between them.

5. Click on the “Unemployment Stats” table. The field in this table “Insured Unemployment Rate” is showing a percentage as a decimal number. This is incorrect. Create a calculated field that converts this value into a proper percentage (e.g. 3.13000 = 3.13% = 0.0313). Paste a screenshot of the formula used and the results in your report.

Formula used, here I know the value is 0.031300 which is equal to our desired result 0.0313 we can fix this later by limiting the column to 4 decimals in data pane while making the dashboard



Your line should now reflect a more accurate forecast that is more aligned with the data. Write 1-2 sentences describing why you believe the initial forecast was flat (hint: there is one major anomaly in the data, think about the impact that may have on forecasting. You can even play around with your forecast to try different options).



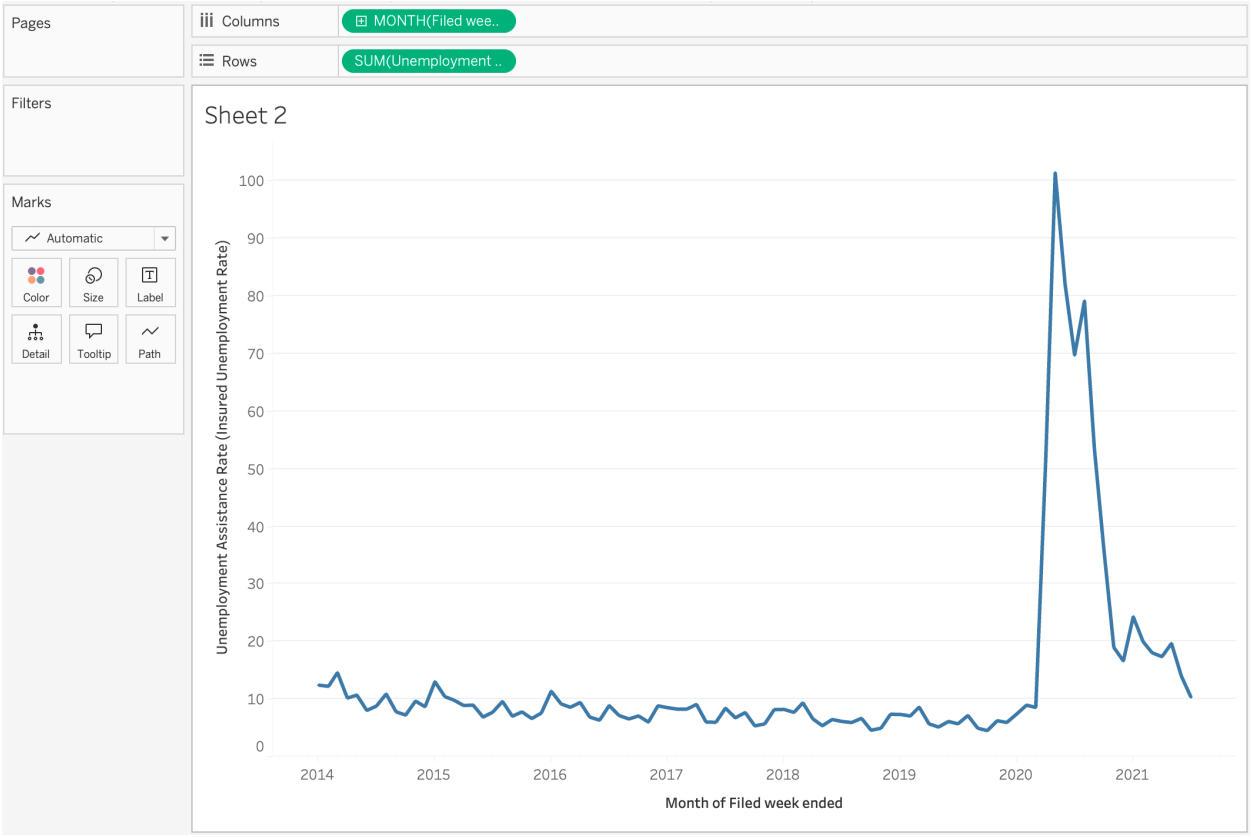
The major anomaly in the data is the sharp decline in employment values around April 2020 to March 2021, corresponding to the COVID-19 pandemic. This sudden drop can significantly affect the forecast by flattening the trend line or introducing irregular patterns in the seasonality, making the default forecasting model less effective.

By switching the trend to "None" and the seasonality to "Additive," the forecast better accounts for this anomaly by focusing on stable, additive patterns in the data post-2020 rather than amplifying the irregularity caused by the pandemic.

Try out your own

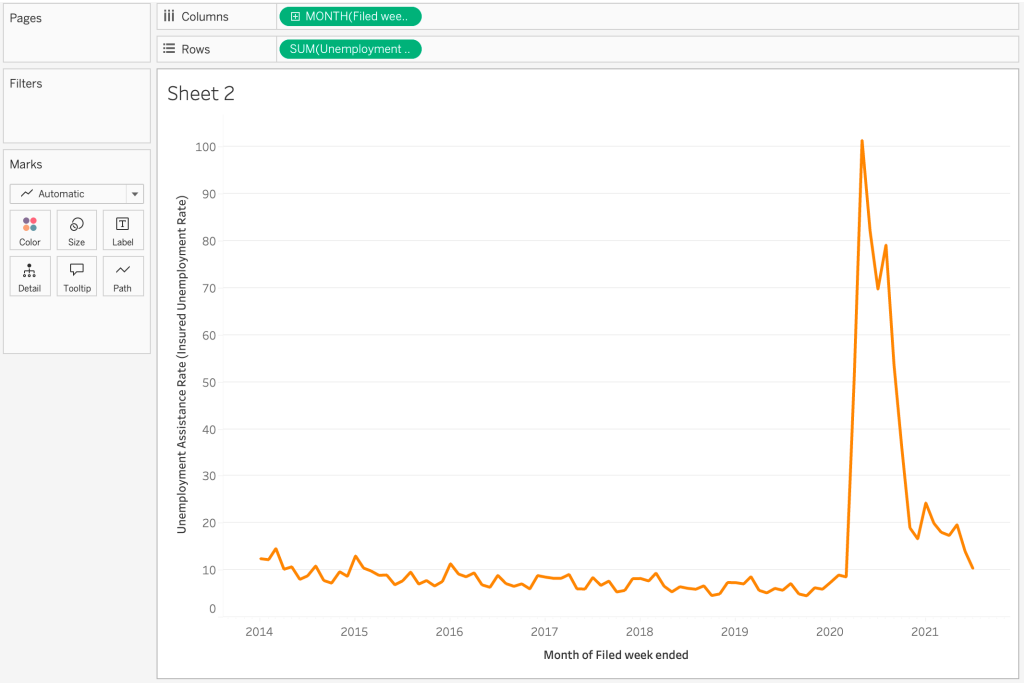
1. Now that you have the basics down, create a second second sheet with a line chart using the “Unemployment Stats” worksheet. It should leverage the following fields: a. Reflecting Week Ended b. Unemployment Assistance rate HINT: Don’t forget to change the date hierarchy to match the time values from the previous section

Paste a picture of the resulting line chart

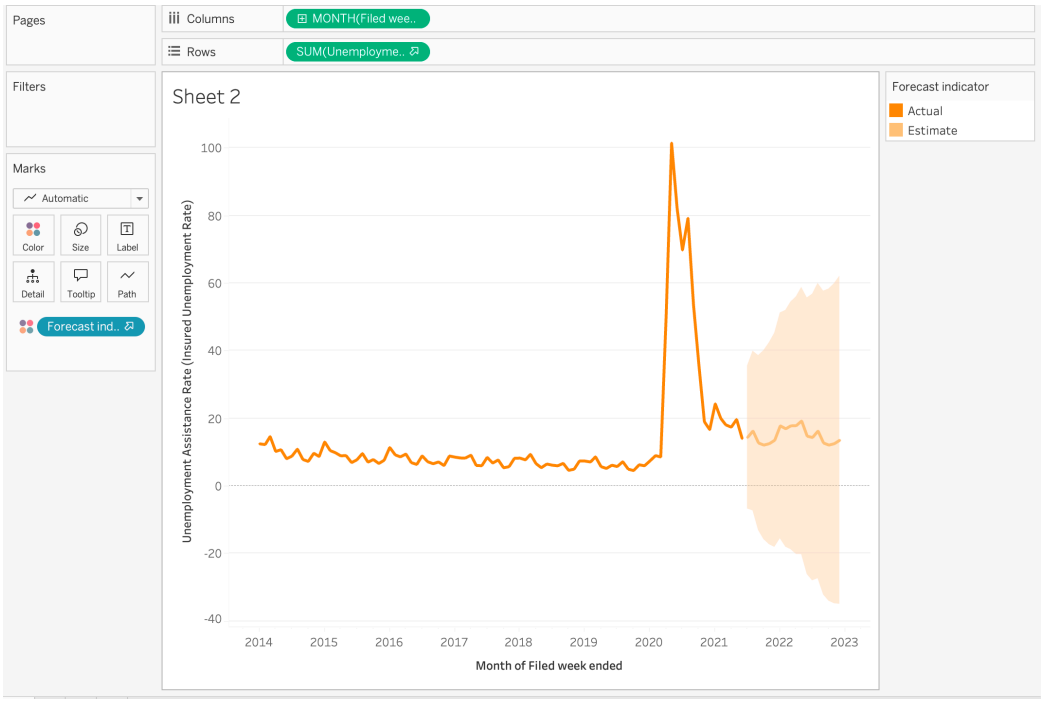


Note: here while doing the assignment I forgot to use calculated filed that were supposed to give us percentage went ahead and took screenshot, but later while creating the dashboard I have changed the filed you can take a look below in main dashboard in Rate intervals on Y-axis.

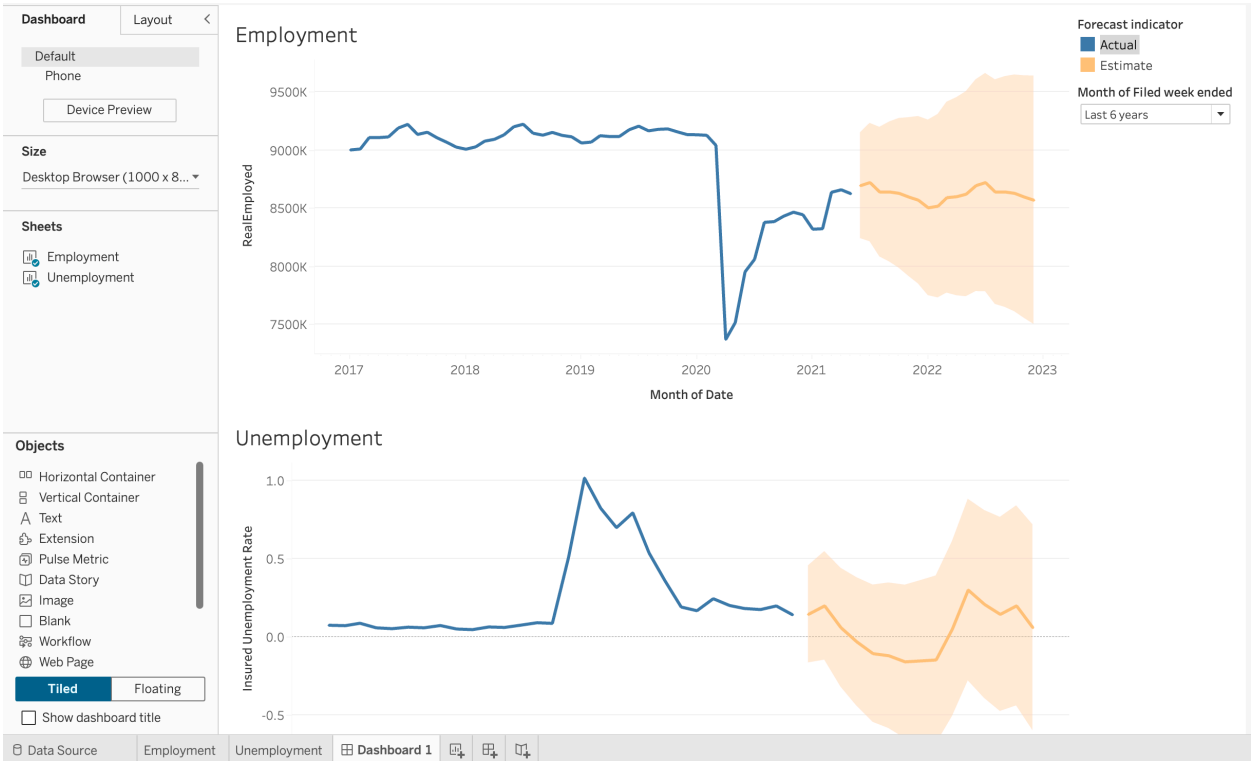
2. Change the color of the line chart to something other than blue. Paste a picture of the new line chart in your report.



3. Create a forecast, setting the model to custom and the season to additive. Paste a picture of the line chart in your report.



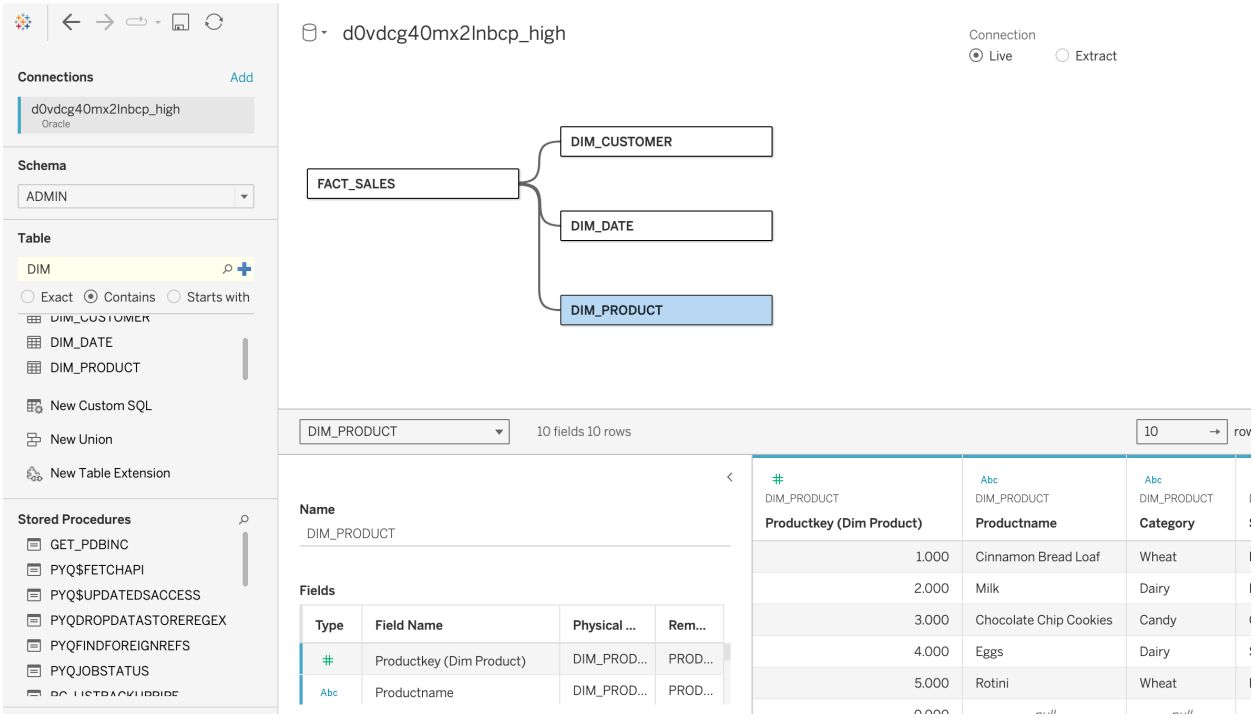
5. Create a Dashboard and add your two sheets to it in a configuration of your choosing. Paste a screenshot of this dashboard in your report.



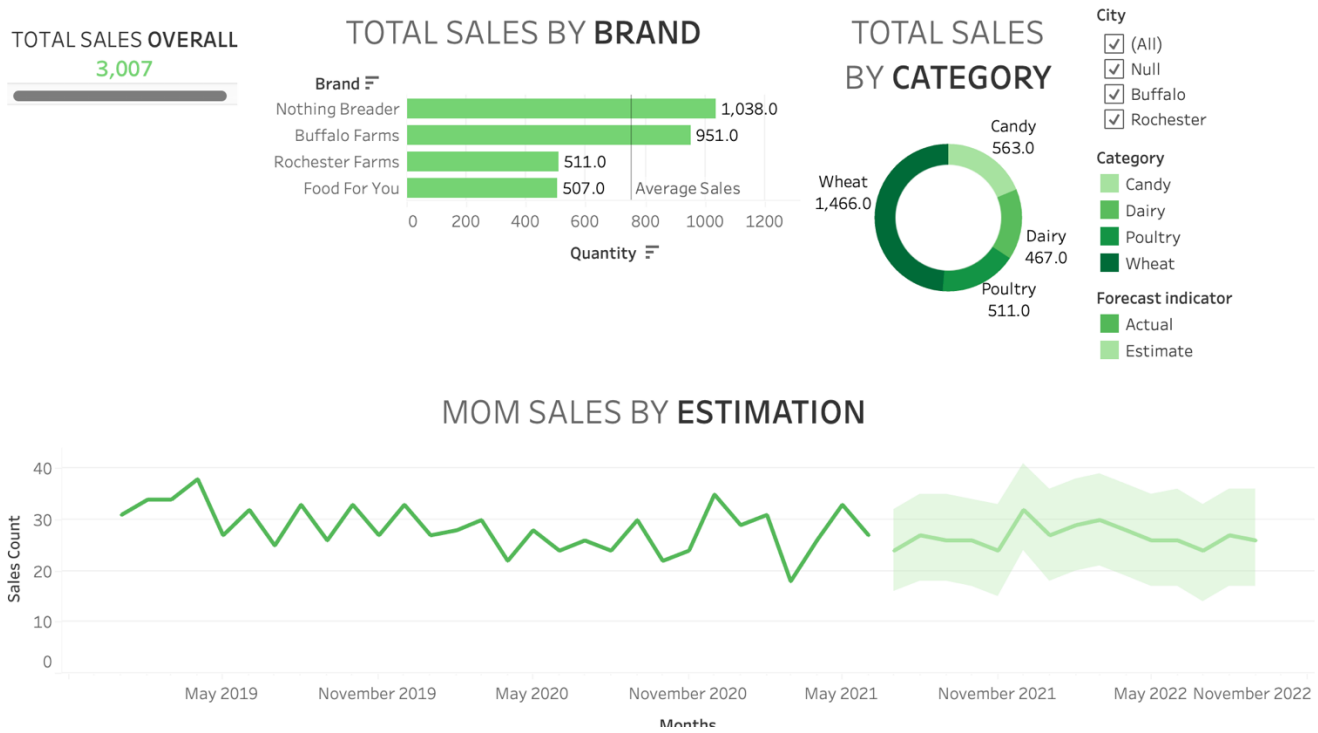
6. Write 2-4 sentences in your report detailing any insights you can draw from these two line charts.

In 2020, COVID-19 pandemic happened that changed everything. You can see this clearly in the charts. Employment took a huge dip, like someone hit the brakes hard, and unemployment shot up quickly, almost like a rocket. After that, things slowly started to get better. People began going back to work, and unemployment started to settle down, but it hasn't quite gone back to the way it was before the pandemic. This shows just how much COVID-19 shook things up in the job market.

Take a screenshot of the resulting relationship diagram and attach it to your report.



TGS Dashboard



This dashboard gives a clear view of sales performance. At the top, the **Total Sales Overall** shows the total quantity sold across all brands and categories. The **Total Sales by Brand** bar chart compares sales by each brand, with an average line to show how they perform against the overall trend. The **Total Sales by Category** donut chart breaks down sales by product categories, making it easy to see which categories are contributing the most. Filters for city and category let you explore specific data. At the bottom, the **Month-over-Month Sales** line chart tracks actual and estimated sales over time, helping to spot trends and patterns. This dashboard is a simple and effective way to analyze sales data.