COMP 430 – PROJECT GROUP 4

E-commerce & Customer Order Analysis Data Warehouse

Final Report

Team Members

Manbir Singh Sidhu

Balkar Singh

Pradiumn

Kartik Karir

Table of Contents

Overview 3
Data Collection 3
- Data Cleaning and Preprocessing 4

ETL (Extract Transform Load) Process 5
Star Schema 7
- Design Process 7

Project Implementation 9
- Addressing the Initial Challenge of Data Sourcing 9
- Building Meaningful Data Relationships 9
- Balancing Local and Cloud Infrastructure with Oracle Database 10
- Leveraging Linux and Docker for Database Hosting 10
- Future Vision of the Project 11
- Choosing DBeaver as the SQL Editor 12
  - Connecting DBeaver to the Docker-Hosted Database 12

Crafting DDL and DML Scripts for the Prototype 13
- Screenshots of the Model and ERD 16
  - Connecting Meta Base to the Oracle Database 16

Visualization 19
  - Warehouse Size Vs Capacity (Staff Size Bubbles) 19
  - Customer representation of the Business in different province 20
  - Visualization of Customer Inquiries regarding the package or service 21
  - Tree Map of capacity of Warehouses 22
  - Shipping Vs Carrier 22

Conclusion 23
Team contributions 23
References 24

Overview

As we approach the final stages of our *E-commerce & Customer Order Analysis Data Warehouse* project, we're proud of the progress we've made—especially in analyzing shipping and logistics performance.

What We've Done:

- Designed a **star schema** in Oracle locally with **Shipping_Fact** as the central fact table.
- Built dimension tables: **Customer_Dim**, **Date_Dim**, **Warehouse_Dim**, and **Product_Dim**.
- Collected and cleaned data from Excel files using **Python, Manual cleaning and Pentaho**.
- Implemented ETL processes using **Pentaho Data Integration**.
- Optimized schema for faster, more efficient querying.

Key achievements include sourcing and cleaning data from Excel files via Python, implementing ETL processes with Pentaho, and optimizing the schema for efficient querying. Challenges like data inconsistencies and ETL setup were resolved through preprocessing and tool configuration.

This report highlights the full journey of our *E-commerce & Customer Order Analysis Data Warehouse* project—from data sourcing to a working visual prototype. Using **Kaggle datasets**, we cleaned and processed data with **Python**, designed a star schema in **Oracle Cloud**, and built ETL pipelines with **Pentaho**, all within a **Linux/Docker** setup. Diagram tools helped us visualize architecture clearly.

Data Collection

We started with a diverse and detailed dataset from Kaggle, a popular site for open datasets.

The data included:

- Customer information (names, locations, etc.
- Product details (categories, prices)
- Order histories (order dates, statuses)
- Transaction records (quantities, shipping info)

This made it great for creating a data warehouse that supports deep business insights and analytics. However, even though the data matched our project requirements, it still required a significant amount of time to refine removing **redundancy**, **handling null values**, and addressing other data quality issues.

Major Platforms used

Data Cleaning and Preprocessing

The raw Kaggle dataset required refinement due to missing values, inconsistencies, and duplicates. A hybrid approach of manual and automated methods ensured data quality:

- **Missing Values**: Python scripts imputed missing data (e.g., prices with medians, categories via patterns).
- **Standardization**: Formats were unified (e.g., dates to YYYY-MM-DD, text to lowercase).
- **Deduplication**: Duplicates were removed using SQL and Python algorithms.
- **Validation**: Manual checks confirmed accuracy and addressed edge cases.

Here is the overview of Excel files (header)

Customer_Dim

Customer_ID	Customer_FirstName	Customer_LastName	Gender	City	Province	Email	Subscription_Status	Customer_Rating	Customer_Calls
1001	John	Smith	Male	Vancouver	British Columbia	john.smith@gmail.com	1	4	4
1002	Alice	Johnson	Female	Toronto	Ontario	alice.johnson@gmail.com	1	2	4
1003	Robert	Williams	Male	Brampton	Ontario	robert.williams@gmail.com	1	3	2
1004	Emily	Brown	Female	Toronto	Ontario	emily.brown@gmail.com	1	2	3
1005	Michael	Davis	Male	Toronto	Ontario	michael.davis@gmail.com	1	4	2

Date_dim

Date_	Date	Day	Month	Day_Numb	Mont	Quart	Quarter_Start	Quarter_End	Ye
20,141,231	12/31/2014	Wednesday	December	365	12	4	undefined	20,141,231	2,014
20,150,101	1/1/2015	Thursday	January	1	1	1	20,150,101	20,150,331	2,015
20,150,102	1/2/2015	Friday	January	2	1	1	20,150,101	20,150,331	2,015

Warehouse_Dim

WarehouseID	Warehouse_Block	Warehouse_Capacit	Location	Temperature_Contro	Staff_Coun
16	B	624	Atlanta	Yes	19
20	C	2539	Atlanta	Yes	31
27	A	1046	Atlanta	No	16

Product_dim

Product_ID	Product_Key	Product_Name	Product_Category	Product_Im	Base_Price	Weight_Clas	Storage_Req
101	P001	Organic Apples	Fresh Produce	High	4.99	Light	Refrigerated
102	P002	Bananas	Fresh Produce	High	2.99	Light	Room Temp
103	P003	Baby Spinach	Fresh Produce	Medium	3.99	Light	Refrigerated

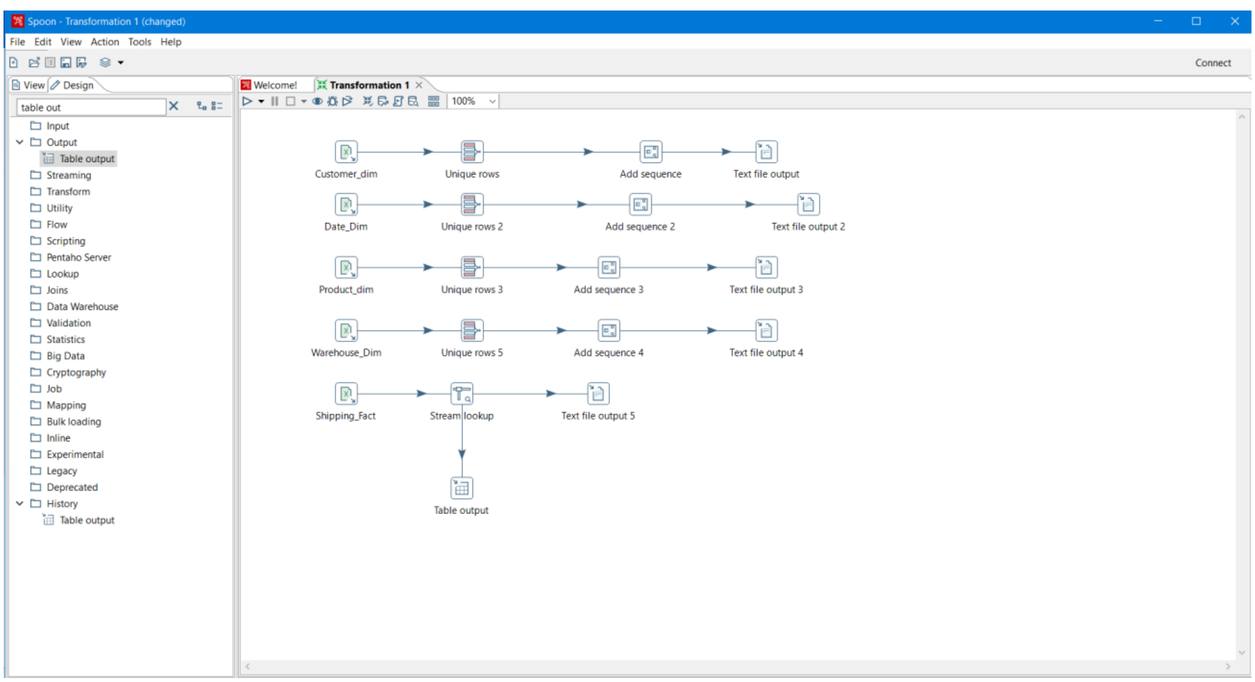
Shipping_Fact

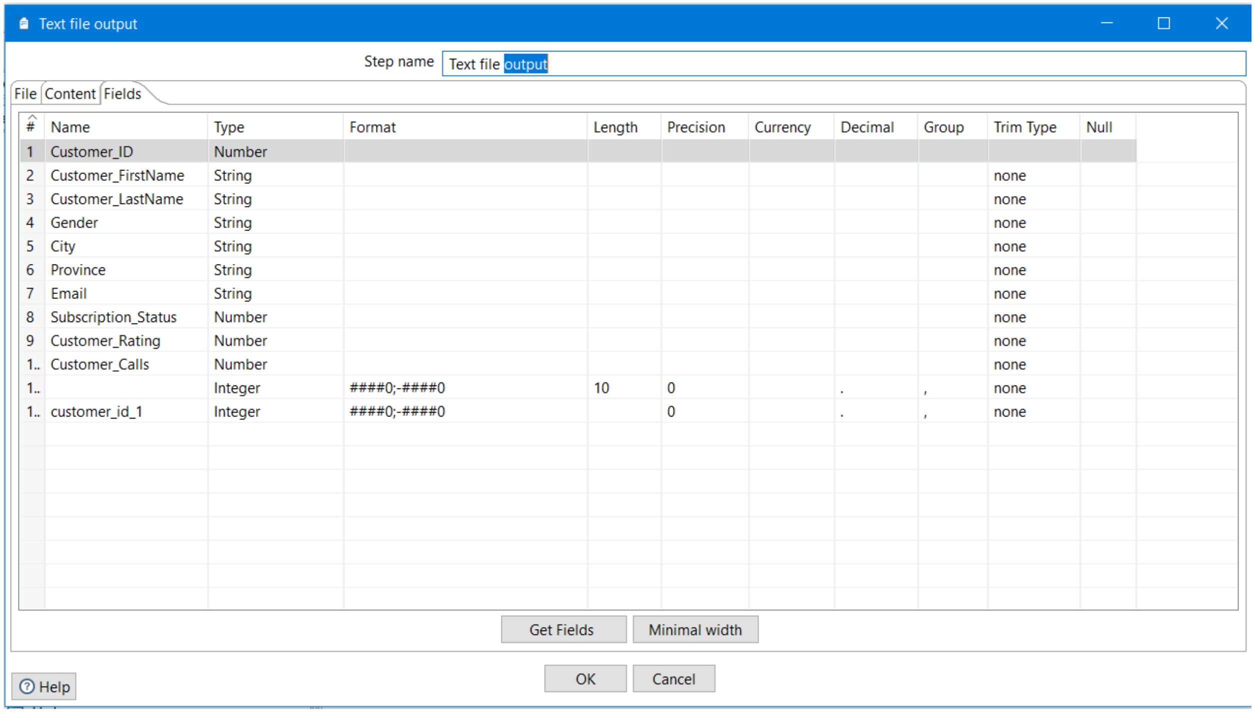
Shipping_ID	Shipping_Quantity	Shipping_Cost	Insurance_Cost	Total_Value	Return_Cost	Estimated_Weight (in g)	Warehouse_ID (FK)	Customer_ID (FK)	Product_ID (FK)	Shipping_Mode	Carrier_Name	Service_Level	Expected_Delivery_Date (FK)	Actual_Delivery_Date (FK)	Transit_Period	Delivery_Location_City	Delivery_Province
1001	4	17	5	33.96	25.5	1200	7	1598	181	Flight	DHL	Standard	20170101	20170102	1	Brampton	Ontario
1002	5	26	5	55.95	39	3100	18	1150	187	Ship	DHL	Standard	20170101	20170105	4	Mississauga	Ontario
1003	3	38	4	80.97	57	3400	5	1183	263	Ship	Canada Post	Standard	20170101	20170108	7	Toronto	Ontario
1004	4	16	2	49.96	24	1200	5	1257	124	Flight	UPS	Express	20170101	20170103	2	Toronto	Ontario

ETL (Extract Transform Load) Process

As mentioned earlier, the data was prepared using Python scripts and manual modifications through an Excel file. We used Pentaho to load the modified Excel file for the extraction process and to output the table data from there.

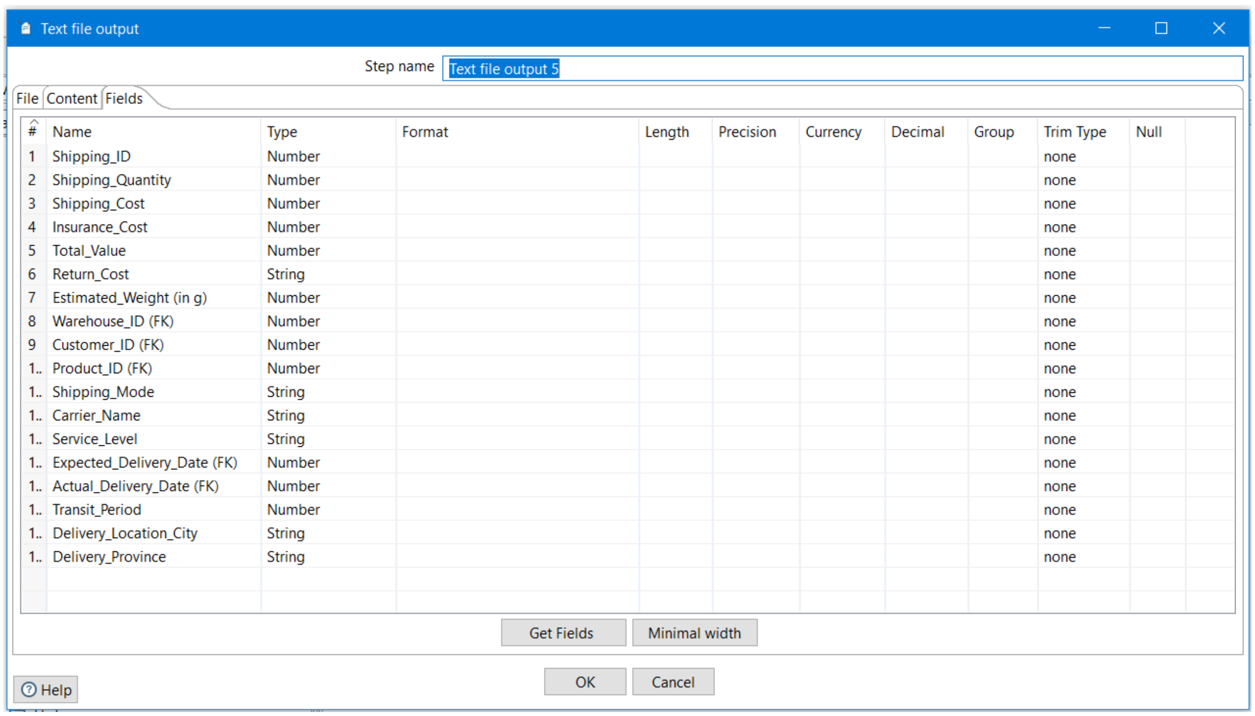
This is a new Pentaho transformation, created after dropping the previous one due to table modifications. We decided not to proceed with **Shipping_Status_Dim**, as the **Shipping_Fact** table already contained sufficient data for our warehouse needs.





#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim Type	Null
1	Customer_ID	Number								
2	Customer_FirstName	String							none	
3	Customer_LastName	String							none	
4	Gender	String							none	
5	City	String							none	
6	Province	String							none	
7	Email	String							none	
8	Subscription_Status	Number							none	
9	Customer_Rating	Number							none	
1..	Customer_Calls	Number							none	
1..		Integer	####0;-####0	10	0		.	,	none	
1..	customer_id_1	Integer	####0;-####0		0		.	,	none	

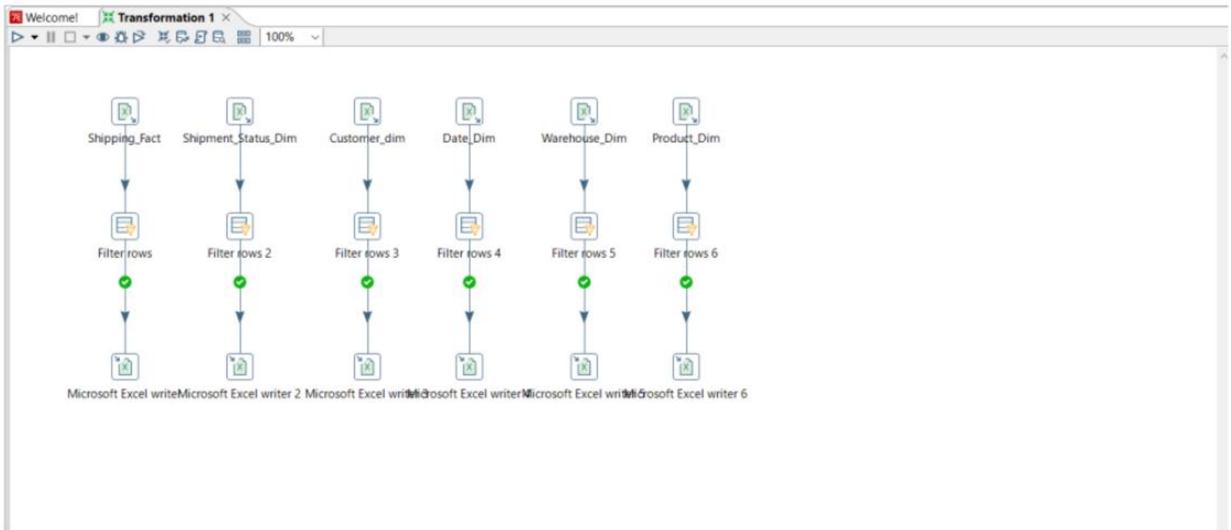
One of the Dimension table in Pentaho with others similarly. ⬆



#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim Type	Null
1	Shipping_ID	Number							none	
2	Shipping_Quantity	Number							none	
3	Shipping_Cost	Number							none	
4	Insurance_Cost	Number							none	
5	Total_Value	Number							none	
6	Return_Cost	String							none	
7	Estimated_Weight (in g)	Number							none	
8	Warehouse_ID (FK)	Number							none	
9	Customer_ID (FK)	Number							none	
1..	Product_ID (FK)	Number							none	
1..	Shipping_Mode	String							none	
1..	Carrier_Name	String							none	
1..	Service_Level	String							none	
1..	Expected_Delivery_Date (FK)	Number							none	
1..	Actual_Delivery_Date (FK)	Number							none	
1..	Transit_Period	Number							none	
1..	Delivery_Location_City	String							none	
1..	Delivery_Province	String							none	

Shipping fact fields using Stream Lookup in Pentaho. ⬆

Old Pentaho Transformation ⬇



From there, we attempted to connect the output table data to the Oracle database, but it failed multiple times. Despite conducting some research to identify the issue, we were unable to resolve it.

Star Schema

The **E-commerce & Customer Order Analysis Data Warehouse** was developed using a **star schema** in **Oracle Cloud** to efficiently manage large volumes of shipping and transactional data. This model centers around a **fact table (shipping_facts)** surrounded by descriptive **dimension tables**: date_dim, product_dim, warehouse_dim, and customer_dim.

Design Process

Fact Table – shipping_facts

- Stores core shipping metrics:
  shipping_quantity, shipping_cost, insurance_cost, total_value, return_cost
- Linked to dimensions via foreign keys:
  warehouse_id, customer_id, product_id, expected_delivery_date_id, actual_delivery_date_id
- Additional logistics fields:
  shipping_mode, carrier_name, transit_period

Dimension Tables

- **Date Dimension (date_dim)**
  Fields: date, month, quarter, year
  Supports trend and seasonality analysis
- **Product Dimension (product_dim)**
  Fields: product_name, product_category, base_price
  Enables product-level sales and shipping insights

- **Warehouse Dimension (warehouse_dim)**
  Fields: warehouse_block, capacity, location
  Used for analyzing distribution efficiency
- **Customer Dimension (customer_dim)**
  Fields: firstname, city, subscription_status
  Useful for customer profiling and segmentation

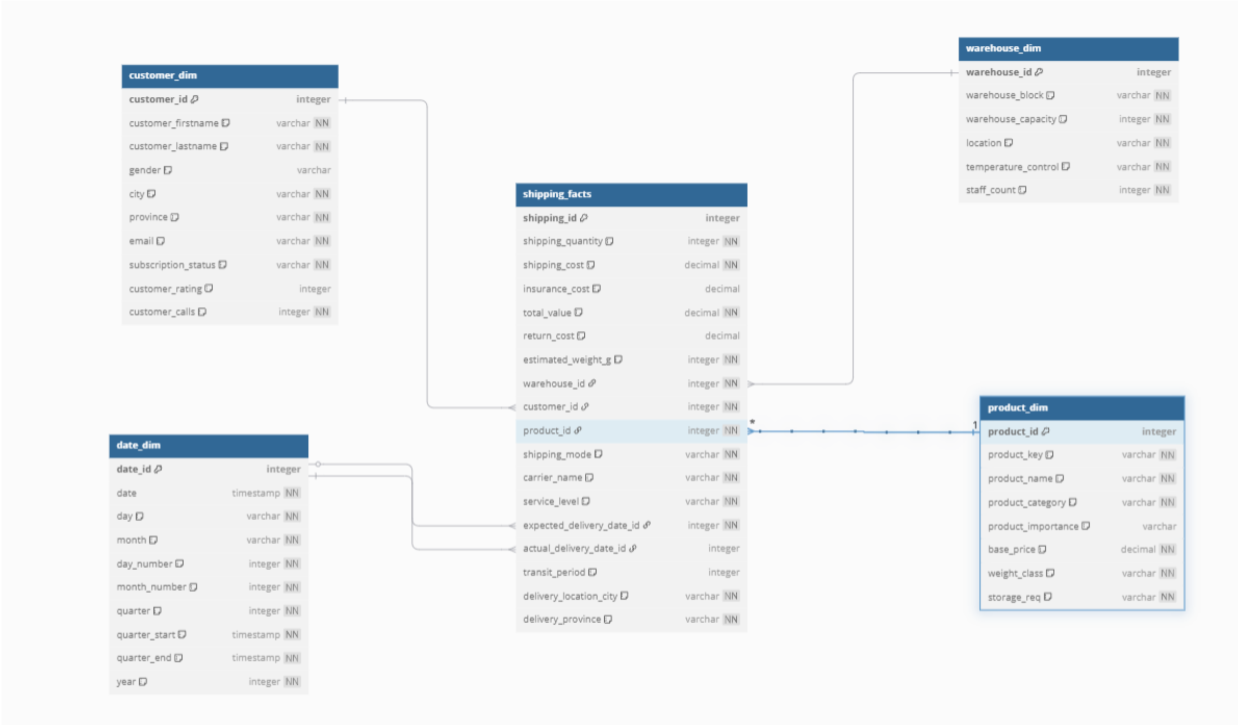
Normalization Strategy

- **Dimension tables** were **denormalized** to reduce joins and improve query performance (e.g., combining day, month, year in date_dim).
- The **fact table** remained **normalized** to maintain data integrity.

Benefits of the Star Schema

- **High Query Performance**
  Simplified structure enables fast SQL queries (e.g., total shipping cost by carrier_name or product_category).
- **Analytical Flexibility**
  Supports slicing and dicing data across time, geography, and product dimensions.
- **Scalability**
  Capable of handling millions of records and allows for easy future expansion (e.g., adding supplier_dim).
- **User-Friendly Design**
  Intuitive schema facilitates data exploration and reporting in tools like **Oracle SQL Developer** and BI platforms.

We have made this ERD Diagram using dbdiagram.com. The diagram has been pasted below:



This ERD design serves as the foundation for our data warehouse implementation, acting as a reference throughout the project and reflecting the overall structure of our system.

Project Implementation

Addressing the Initial Challenge of Data Sourcing

The first challenge was finding **reliable and relevant data** for the warehouse. We chose **Kaggle** as our data source because it offers **well-documented and diverse datasets**.

To process the data, we used **Python** with libraries like **Pandas**, **NumPy**, and **SciPy** for cleaning and analysis.

We wrote **custom scripts** to clean, normalize, and structure the data for warehouse use.

These scripts were uploaded to **GitHub** for version control, transparency, and future reuse.

This approach helped set a **strong foundation** for the project and ensured the data was ready for the next steps. I have attached the GitHub Repository here: https://github.com/SidhuManbirSingh/Database_430.git

Building Meaningful Data Relationships

With the raw data prepared, the next critical step was to establish meaningful connections that would form the backbone of the data warehouse's relational structure. For this, I once again relied on **Python and Pentaho**, using it to simulate and relationships such as **Primary Keys (PK)** and **Foreign Keys (FK)** across datasets. We used **Pentaho** to clear the redundant values from the table. This phase was pivotal in transforming disparate datasets into a cohesive system, laying the groundwork for efficient querying and analysis in the data warehouse.

Balancing Local and Cloud Infrastructure with Oracle Database

At this point, with the data model taking shape, I had to decide where and how to host the data warehouse.

Initially, I considered using **Oracle Database Cloud**, but hosting locally would offer **faster access** and **lower latency**, while a cloud-based solution would provide **better scalability** and **easier access**. We decided to host the oracle database locally.

After considering the project's long-term goals, I chose an IAAS based approach for scalability to handle future growth. To balance local speed with cloud flexibility, I followed best practices for cloud deployment, ensuring seamless scaling, performance, and cost efficiency.

Leveraging Linux and Docker for Database Hosting

To bring the data warehouse to life, I selected **Linux** as the operating system, drawn by its stability, open-source nature, and widespread adoption in enterprise environments.

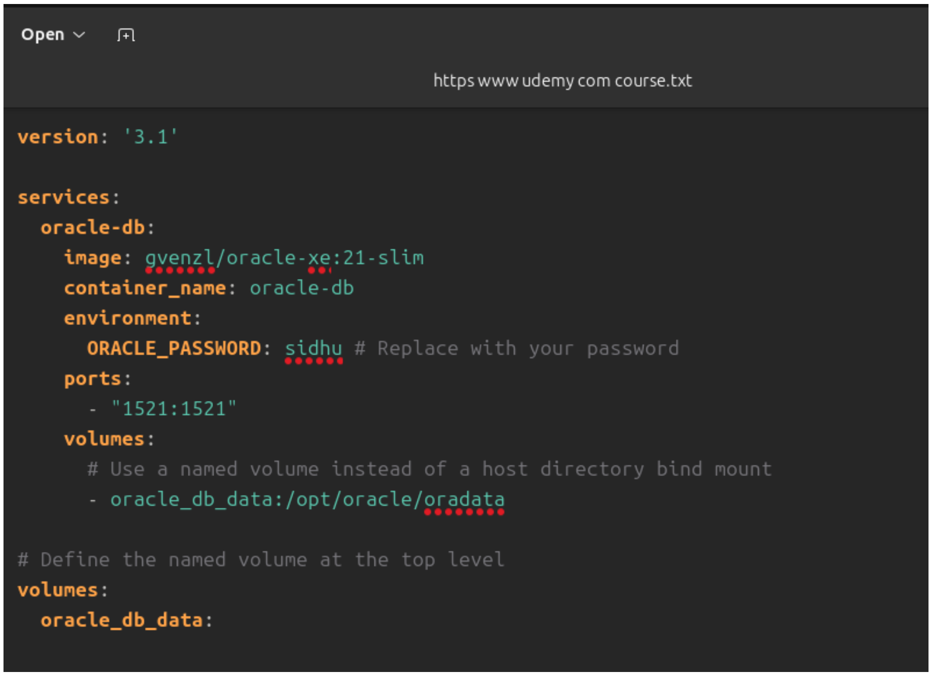
We decided to host the **Oracle Database** inside a **Docker container**, a decision was good for us in terms of stability and performance.

Docker's containerization ensured the database worked consistently across all environments, simplifying setup and reducing configuration errors.

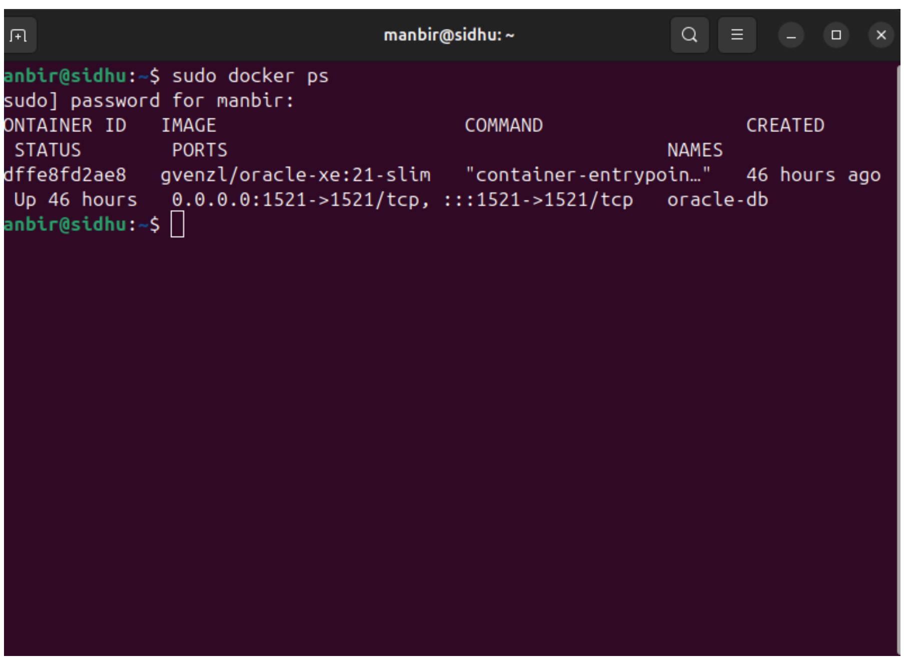
Additionally, Docker enabled easy scalability by adding instances as data or query loads grew. It also allowed running the Oracle Database on modest hardware during prototyping, with the option to scale in the cloud later, simplifying deployment and maintenance. (Reference: [Get started | Docker Docs])

- To implement the Oracle Database in docker, I first of all pulled the docker image and I wrote the **docker-compose.yaml** to get started.

Context:
You typically don't need to pull the image manually. However, you'll need to provide login credentials in the terminal, which should be handled carefully, as your password will be stored unencrypted on your machine



- I used Visual Studio Code to get started with Oracle Database initialization.
- After running docker-compose up -d
- After some time, I got everything set up.



Future Vision of the Project

If we were to implement this setup on an actual Linux server, it would clearly demonstrate its scalability and real-world applicability. Running this configuration in a production-like environment would not only showcase the robustness and efficiency of Linux-based systems but also highlight how easily the solution can scale to support larger workloads and multiple users.

- Adding a second Docker container to replicate the first would increase redundancy and ensure future-proofing.

- This would improve system reliability and make it easier to handle higher traffic and data loads.
- We believe this approach would provide a solid foundation for scaling the solution in real-world environments.

We chose **Oracle IaaS over SaaS** due to the **greater flexibility, control, and customization** it offers. IaaS provides full control over infrastructure—VMs, storage, networking, and OS—allowing us to meet specific technical and security needs.

Unlike SaaS, which is limited in scope, IaaS supports custom applications, frameworks, and configurations, making it ideal for development, testing, and specialized workloads. It's also more cost-effective in multi-service environments and better suited for strict security and compliance requirements.

Choosing DBeaver as the SQL Editor

To interact with our data warehouse, we chose DBeaver, a cross-platform SQL editor and database management tool, for its powerful features and flexibility.

Advantages of Using DBeaver:

- Cross-platform compatibility: Works seamlessly on Linux, Windows, and macOS.
- User-friendly interface: Simplifies SQL queries, schema management.
- Multi-database support: Connects easily to Oracle and other databases, even in a containerized setup.

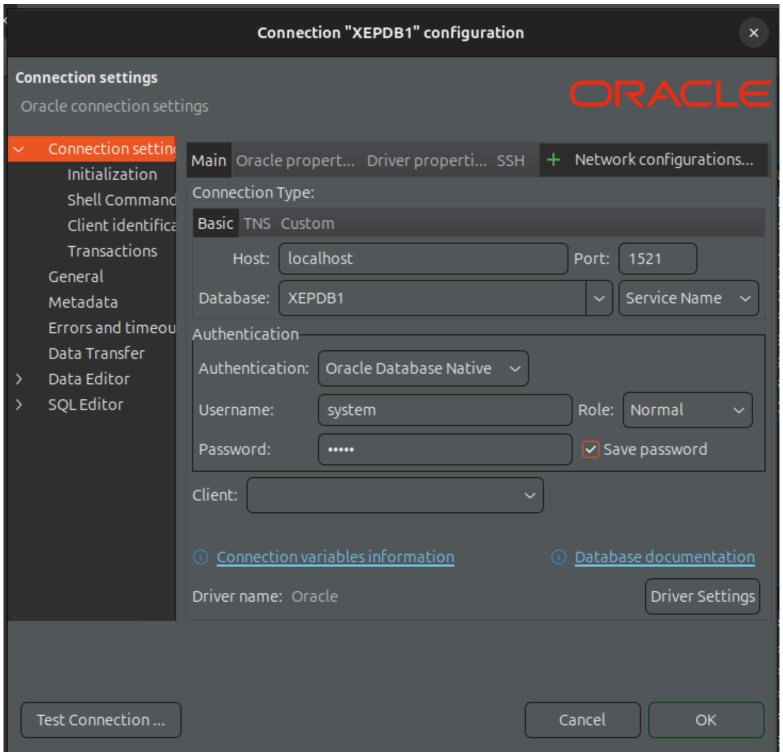
DBeaver's **ERD visualization tools** enabled me to refine and display the data model directly within the application. These advantages made DBeaver an indispensable ally in managing the data warehouse, from writing queries to performing administrative tasks.

Connecting DBeaver to the Docker-Hosted Database

To connect DBeaver to the Docker-hosted Oracle Database, I configured the Edit Connection settings by specifying the localhost and mapping the exposed port (1521) from Docker.

Next, I entered the **database credentials** (username and password) configured within the Oracle instance, ensuring secure access.

I also verified that the **Oracle JDBC Driver** was properly installed and selected in DBeaver, as this facilitated communication between the application and the database.



- After configuring the parameters, I tested the connection and resolved any issues like port conflicts or firewall restrictions.
- This successful setup allowed me to query the database, manage its schema, and visualize the ERD directly in DBeaver, streamlining the development process.

Crafting DDL and DML Scripts for the Prototype

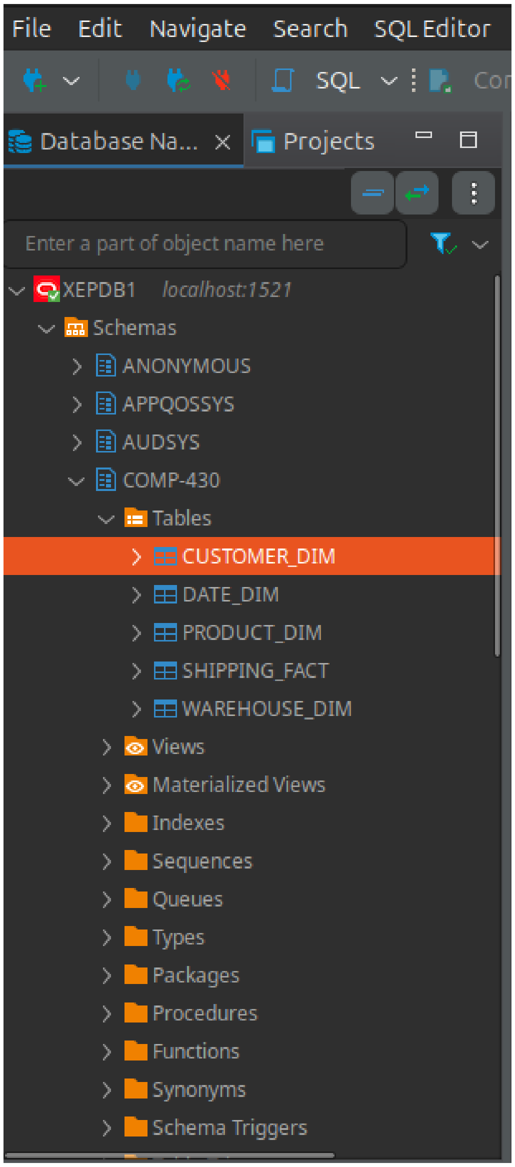
The structure and content of the data warehouse were defined using DDL and DML scripts. The DDL scripts set up the schema—tables, columns, constraints, and indexes—ensuring the database followed the ERD and supported efficient querying.

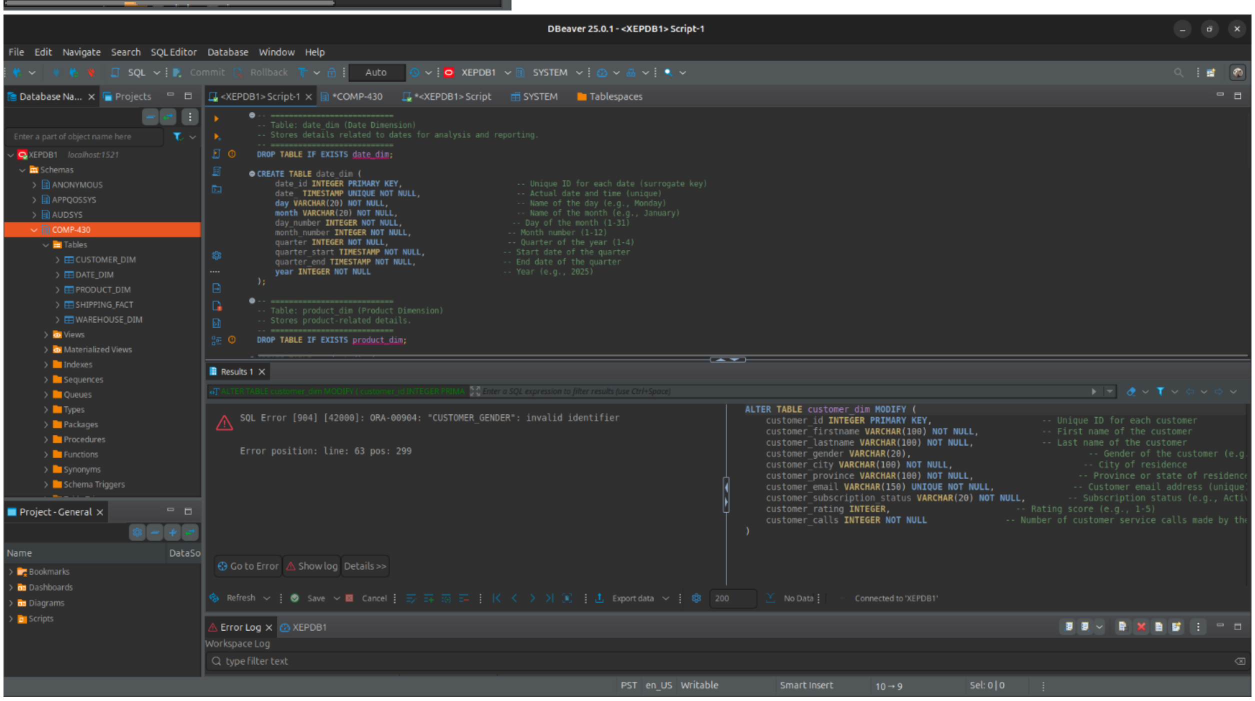
Python was used to manage the creation of DML and DDL scripts, ensuring accuracy and consistency.

- I faced the challenge of converting Excel tables into a relational database structure. To solve this, I developed a Python script that extracted data from Excel and converted it into SQL queries.

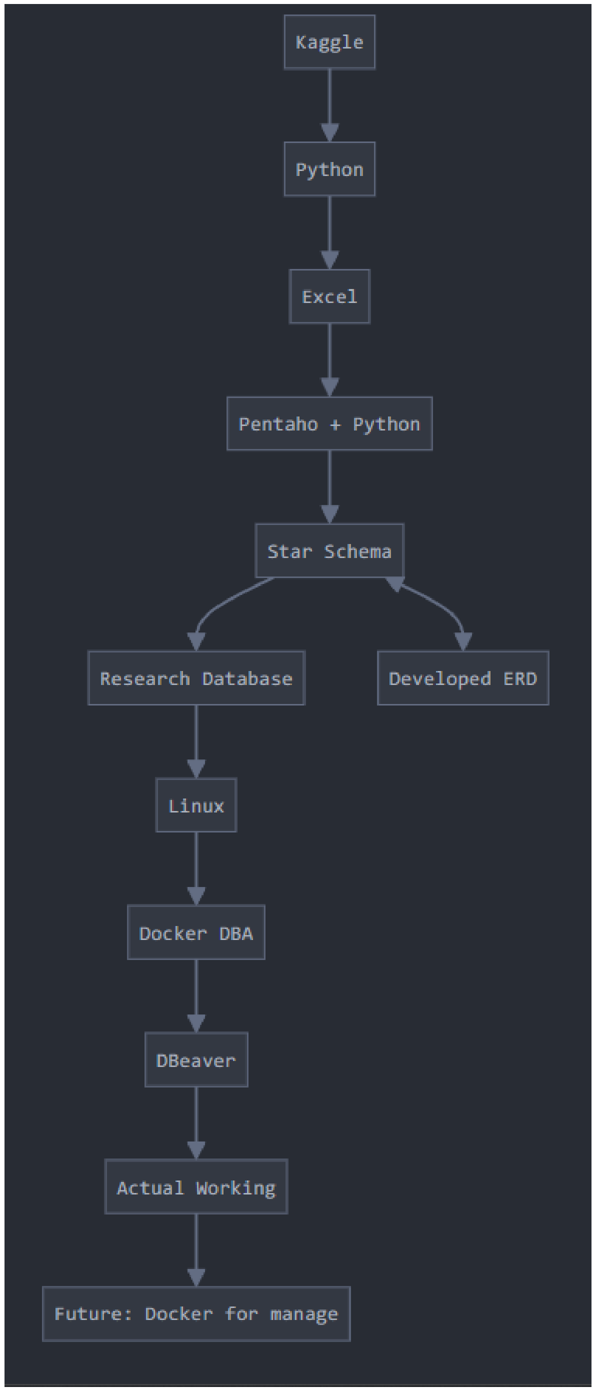
The queries were executed in the Oracle Database through DBeaver

- While I ran the process manually for this project, the script can be automated using a **crontab job** for future use. Here are the screenshots of the successful executions in DBeaver.


File Edit Navigate Search SQL Editor
SQL
Database Na...
Projects
Enter a part of object name here
XEPDB1 localhost:1521
Schemas
ANONYMOUS
APPQOSSYS
AUDSYS
COMP-430
Tables
CUSTOMER_DIM
DATE_DIM
PRODUCT_DIM
SHIPPING_FACT
WAREHOUSE_DIM
Views
Materialized Views
Indexes
Sequences
Queues
Types
Packages
Procedures
Functions
Synonyms
Schema Triggers

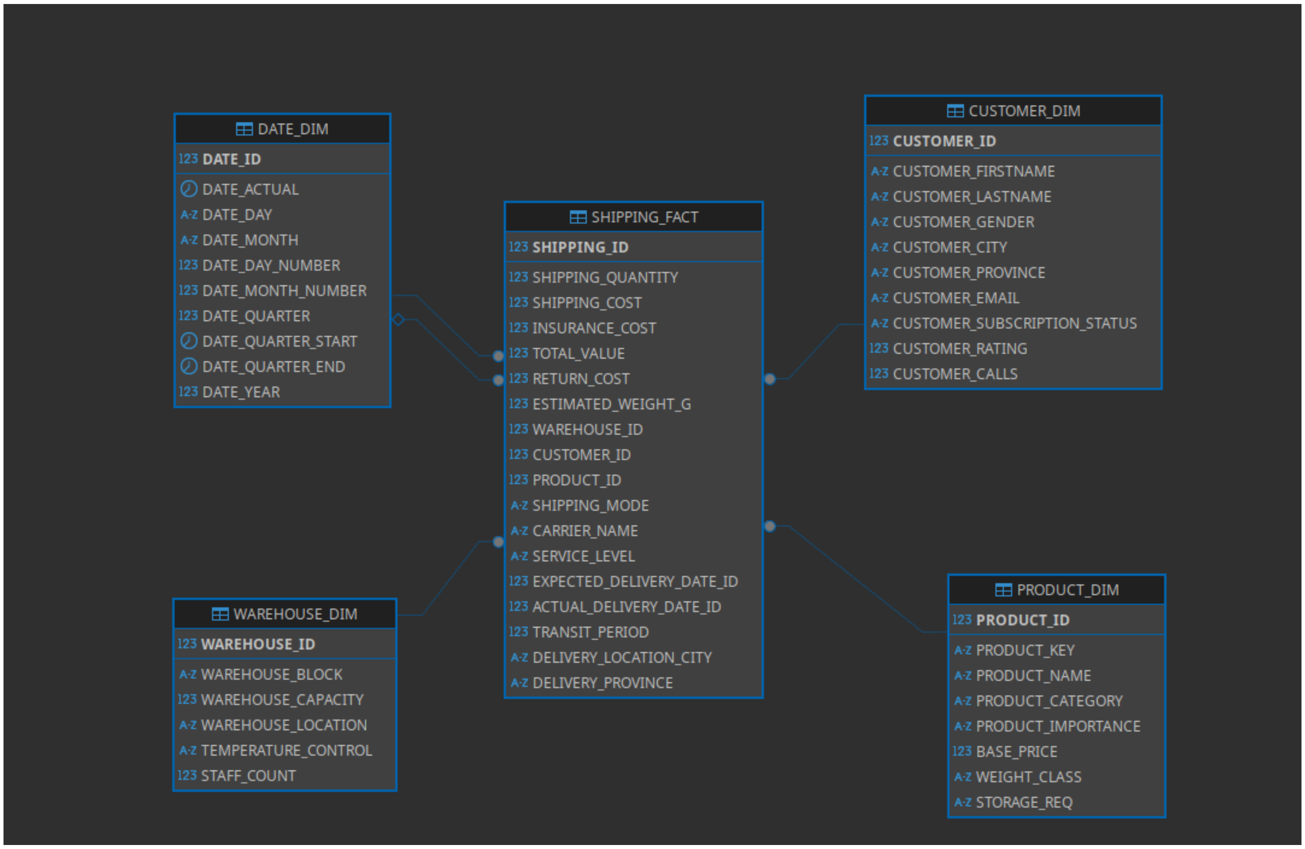

DBeaver 25.0.1 - <XEPDB1> Script-1
Results 1
SQL Error [904] [42000]: ORA-00904: "CUSTOMER_GENDER": invalid identifier
Error position: line: 63 pos: 299
ALTER TABLE customer_dim MODIFY (
customer_id INTEGER PRIMARY KEY, -- Unique ID for each customer
customer_firstname VARCHAR(100) NOT NULL, -- First name of the customer
customer_lastname VARCHAR(100) NOT NULL, -- Last name of the customer
customer_gender VARCHAR(20),
customer_city VARCHAR(100) NOT NULL, -- City of residence
customer_province VARCHAR(100) NOT NULL,
customer_email VARCHAR(150) UNIQUE NOT NULL,
customer_subscription_status VARCHAR(20) NOT NULL,
customer_rating INTEGER, -- Rating score (e.g., 1-5)
customer_calls INTEGER NOT NULL
)
Go to Error
Show log
Details >>
Connected to 'XEPDB1'

Now I will be attaching a screenshot which will be showcasing our progress in the project.



Screenshots of the Model and ERD

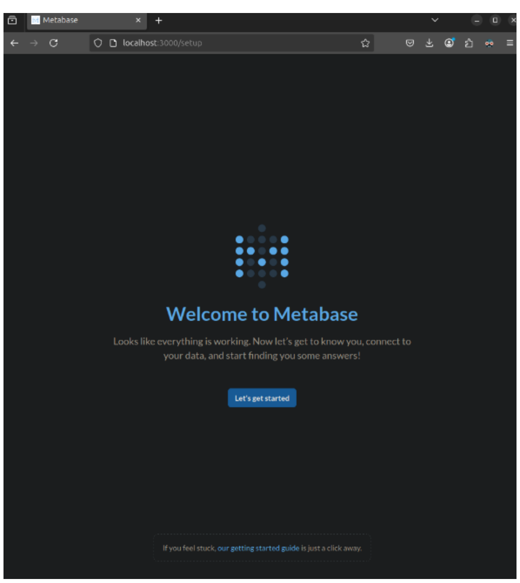
To showcase the success of the implementation, I captured screenshots of the data model and fully functional ERD using DBeaver's built-in visualization tools.



By using this application, we were able to replicate the initial foundation and clearly present the table structures, relationships, and constraints of our data warehouse architecture.

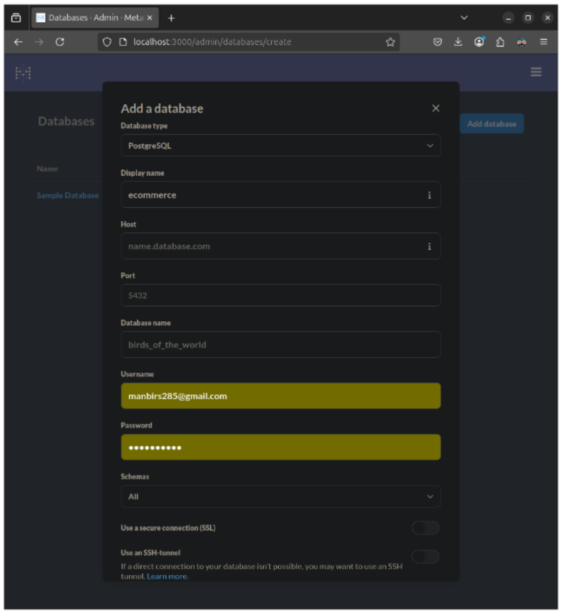
- It is the working model of the ERD we imagined at first, but this is a working model.

Connecting Meta Base to the Oracle Database

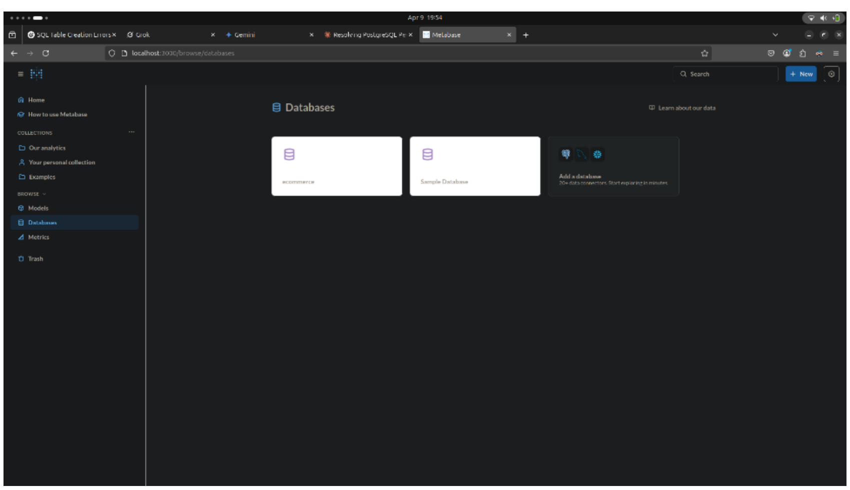


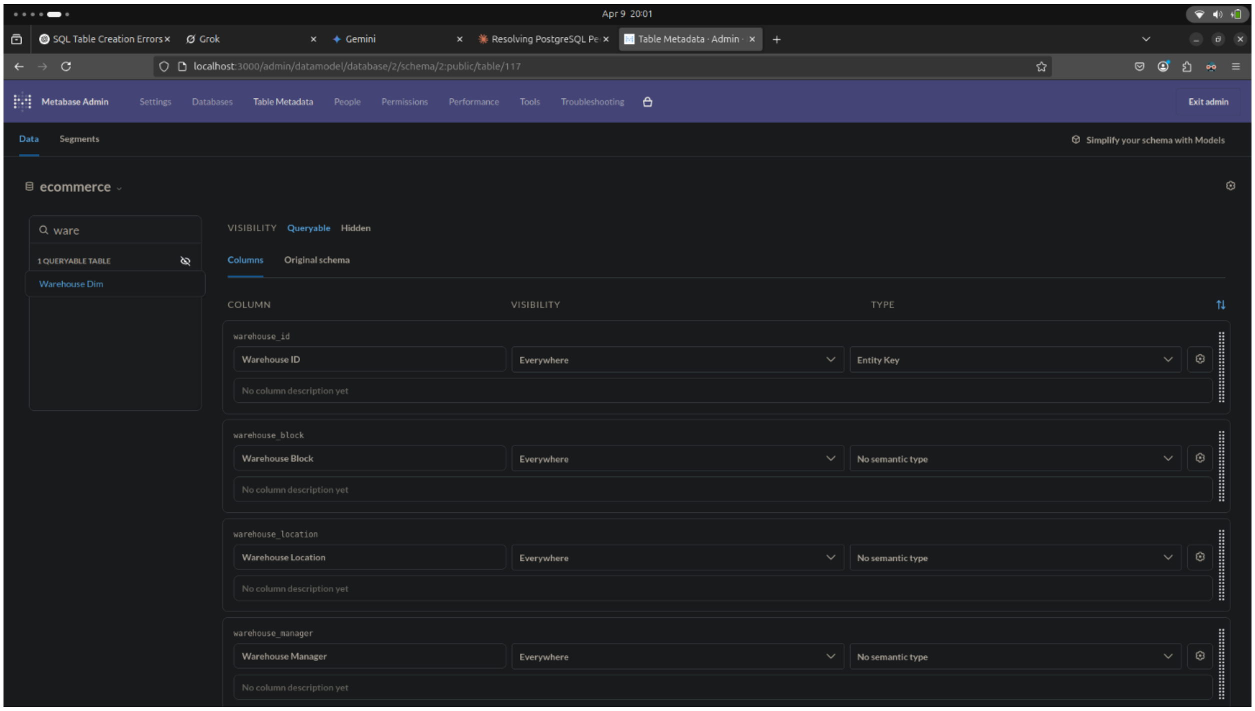
To enhance our data warehouse's analytical capabilities and provide a user-friendly interface for business intelligence, we integrated Metabase, an open-source BI tool, with our Docker-hosted Oracle Database.

The connection process began by ensuring the Oracle Database was accessible via the exposed port (1521) in our Docker setup. In Metabase, we configured a new database connection by specifying the host (localhost), port, database name, and credentials set up in the Oracle instance.



We also installed the Oracle JDBC driver in Metabase to enable seamless communication. After testing the connection and resolving minor issues like network accessibility and driver compatibility,





Benefits and Challenges of Metabase Integration

The integration of Metabase brought several advantages, including its ability to generate shareable dashboards, which streamlined collaboration and reporting. It also supported ad-hoc analysis, enabling us to quickly respond to hypothetical business questions, such as identifying the most cost-efficient shipping carriers or tracking delivery delays by region.

However, we encountered challenges, such as optimizing query performance for large datasets and ensuring proper user access controls within Metabase.

Future Potential with Metabase

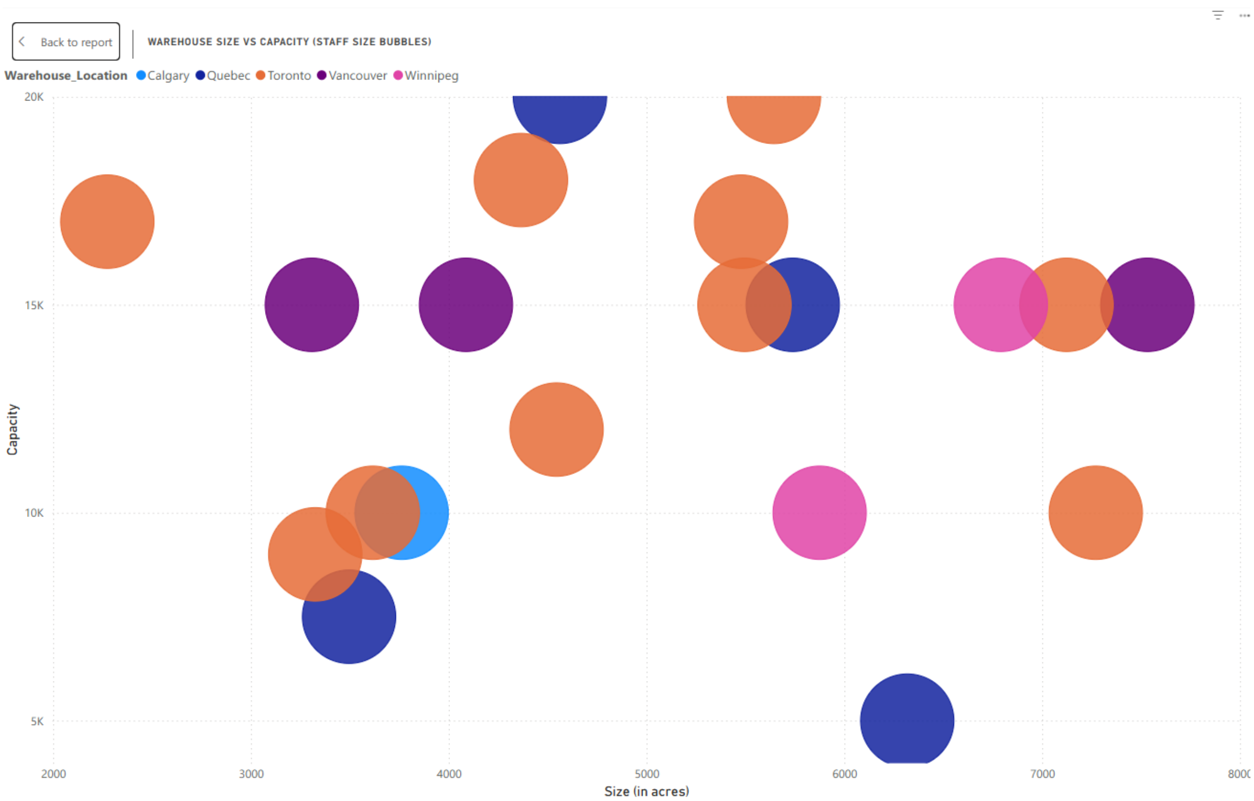
Looking ahead, Metabase's integration opens doors for future enhancements, such as embedding dashboards into external applications or automating reports for stakeholders. Its lightweight deployment within our Linux/Docker environment aligns with our scalability goals, allowing us to potentially expand the system to handle larger datasets or additional data sources. By combining Metabase with our Oracle Database, we've created a powerful analytical layer that not only validates our data warehouse design but also positions it as a practical tool for real-world e-commerce analytics.

Visualization

The following section includes visual representations that show key stages of the project, such as data extraction, transformation, and integration within the database.

These visuals**, created using Power BI, are accompanied by brief explanations to highlight their significance and role in the project.**

Warehouse Size Vs Capacity (Staff Size Bubbles)

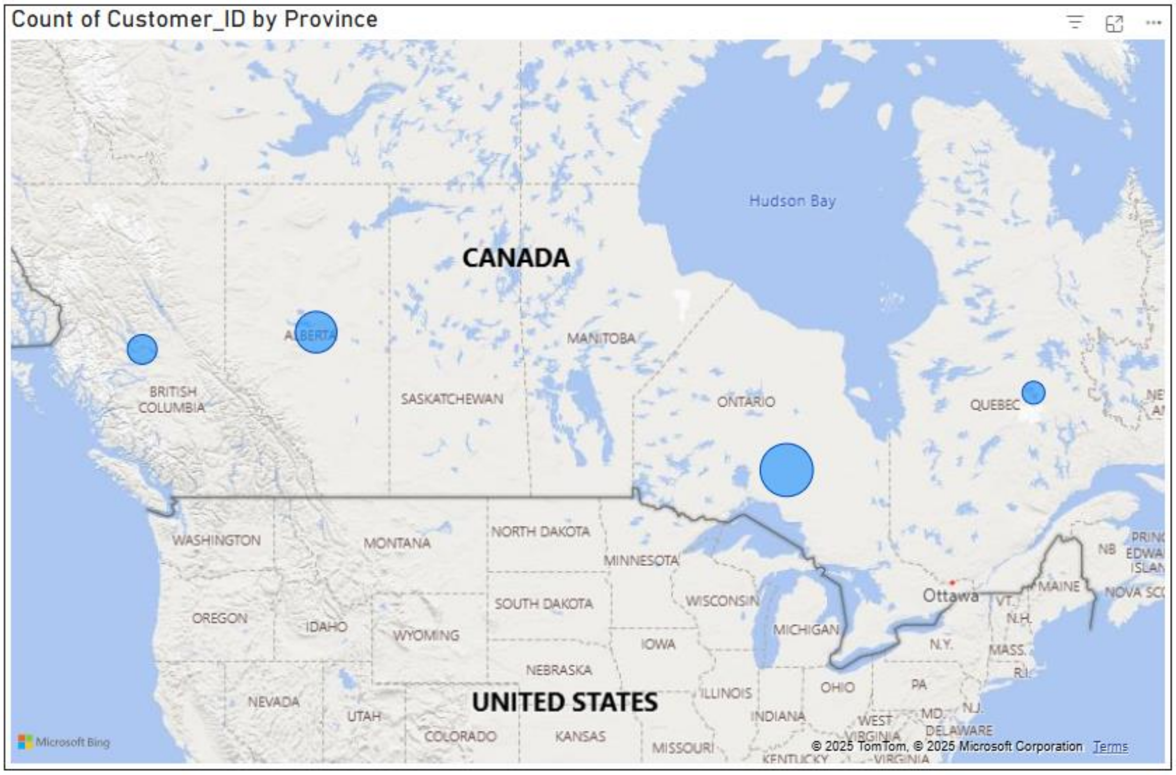


This bubble chart illustrates the relationship between warehouse size (in acres) and capacity across various locations, with each bubble representing a specific warehouse.

The size of the bubbles corresponds to staff size, while the color differentiates between locations such as Calgary, Quebec, Toronto, Vancouver, and Winnipeg.

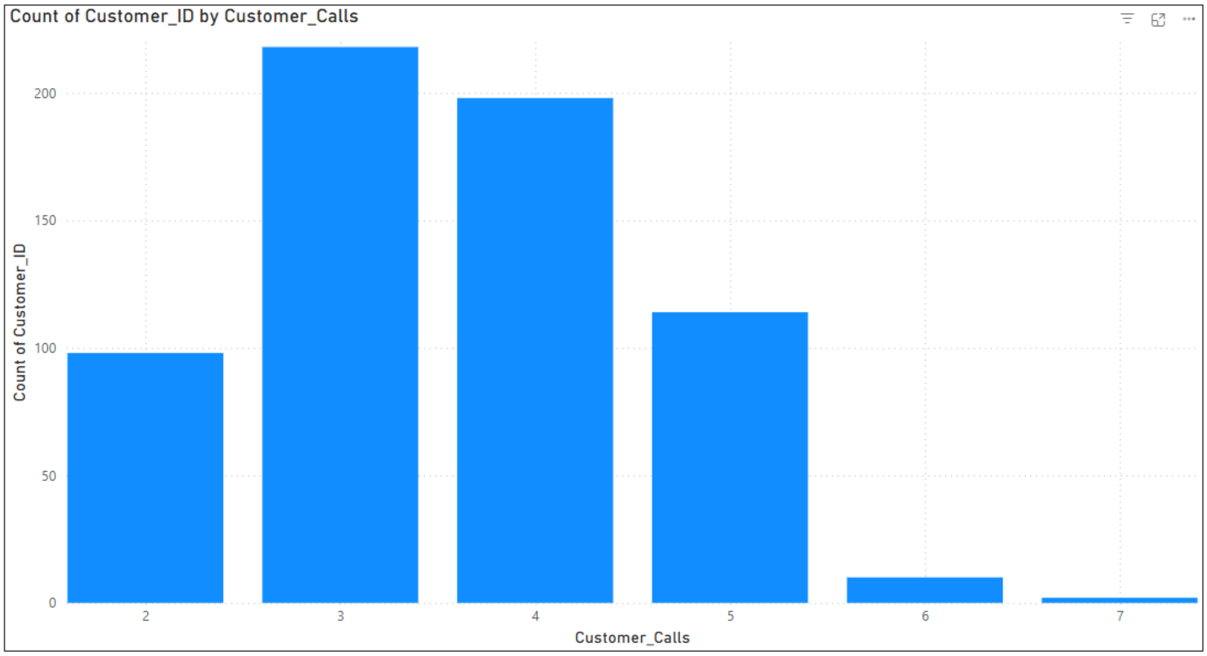
This visualization provides an intuitive overview of how warehouse scale and capacity vary by region and how staffing levels compare across different facilities.

Customer representation of the Business in different province



- Each blue bubble represents the volume of customers in a specific province.
- The size of the bubble indicates the relative number of customer entries
- Provinces like Ontario, Alberta, and Quebec show higher activity, reflecting larger customer concentrations.
- This geographical distribution helps identify key market areas and potential regions for targeted business strategies.

Visualization of Customer Inquiries regarding the package or service

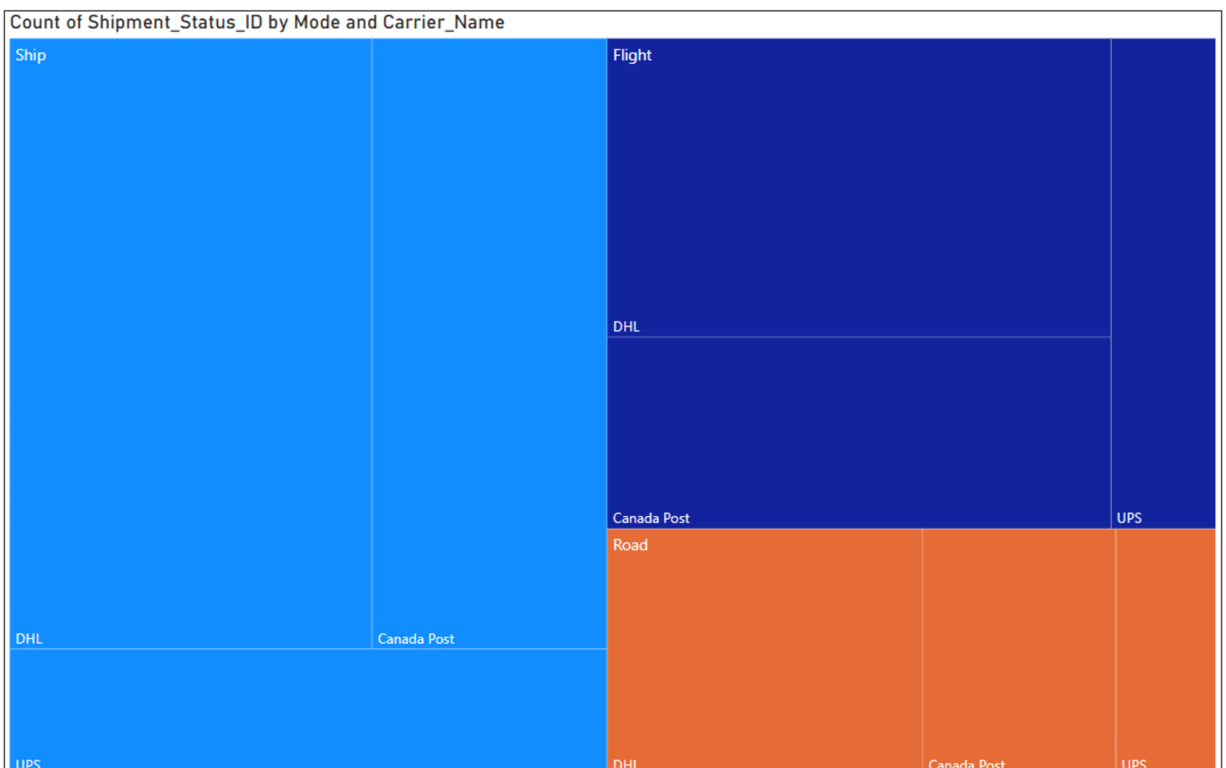


- Each **bar** shows the actual volume of calls received by Customer Service Representatives.
- This information helps identify **anomalies** in call volume.
- Managers can use this data to **set benchmarks** for their staff members.
- Striving for a balanced call volume is crucial—**too few or too many calls** can indicate operational issues.

Tree Map of capacity of Warehouses

Each colored block represents a unique warehouse, with the size of the block indicating the total capacity. Larger blocks signify higher capacity values, making it easy to compare the contribution of each warehouse to the overall capacity. This visualization aids in quickly identifying high-capacity warehouses and understanding how warehouse size correlates with storage potential.

Shipping Vs Carrier



The treemap shows "Ship" is the dominant shipment mode. DHL handles the largest share of shipments in both "Ship" and "Flight" modes.

Canada Post has a significant presence in "Ship" and "Road" transport. UPS handles a smaller share in "Ship" but is notable in "Flight" and "Road".

Conclusion

Our E-commerce & Customer Order Analysis Data Warehouse project has been a valuable learning experience. Through this project, we gained hands-on experience with key concepts in data warehousing, including data extraction, transformation, and loading (ETL), as well as designing a star schema for efficient data storage and querying.

Key achievements:

- We designed a **star schema** that improved query performance and made analysis easier.
- We used **Pentaho** for ETL processes, transforming and loading data into the Oracle database.
- **Data cleaning** was crucial, as we handled missing values, duplicates, and standardization using Python.
- We connected **DBeaver** to the Oracle database, enabling smooth schema management and SQL query execution.

Our experiment with this data warehouse helped us understand the importance of structuring data effectively for business analysis. We learned how to create relationships between different data sets (like customer, product, and shipping data) and how these relationships support better decision-making.

This project also deepened our understanding of how to scale and manage data warehouses using **Docker** and **Oracle Database**, ensuring the system could grow with future needs. The visualizations we created using **Power BI and Metabase** helped us gain insights into customer behavior and operational efficiency.

Team contributions

Member	Contributions
Manbir Singh Sidhu	Built ETL scripts, resolved data transformation challenges, automated Using Python, Researched the Viable Data Sources, Data Integration
Balkar Singh	Designed star schema, UML Diagram, Pentaho ETL Process, Excel Data refining, Researched for data design and integration & Documentation.
Kartik Karir	Documentation, Formatting the Report, Visual Creations
Pradiumn	Documentation, Formatting the Report, Research over Data Integration

References

Aravind. (2013, December 25). *Pentaho Data Integration - Configure Oracle JDBC Connection*. Knowpentaho.com. https://www.knowpentaho.com/2013/12/pentaho-data-integration-configure.html

Developing Python Applications for Oracle Database. (2018). Oracle.com. https://www.oracle.com/database/technologies/appdev/python/quickstartpythononprem.html

Friederichs, N. (2024, February 5). *Setting Up a PostgreSQL Environment in Docker: A Step-By-Step Guide*. Medium. https://medium.com/@nathaliafriederichs/setting-up-a-postgresql-environment-in-docker-a-step-by-step-guide-55cbcb1061ba

How to connect dbeaver with docker instance of mysql locally. (2021, July 5). Stack Overflow. https://stackoverflow.com/questions/68258754/how-to-connect-dbeaver-with-docker-instance-of-mysql-locally

https://community.fabric.microsoft.com/t5/user/viewprofilepage/user-id/320. (2015, December 16). *Configuring Power BI Connectivity to PostgreSQL Database*. Microsoft.com. https://community.fabric.microsoft.com/t5/Power-BI-Community-Blog/Configuring-Power-BI-Connectivity-to-PostgreSQL-Database/ba-p/12567

Sharma, M. (2018, October 12). *Star Schema in Data Warehouse modeling - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/star-schema-in-data-warehouse-modeling/

Star Schema Foundations. (2025). Pluralsight.com. https://www.pluralsight.com/courses/star-schema-foundations?clickid=&utm_source=bing&utm_medium=paid-search&utm_campaign=upskilling-and-reskilling&utm_term=ssi-na-bing-ca-dynamic&utm_content=free-trial