



Milestone

Present by: Harshitha Mayya

AWS (Amazon Web Services) DB Login

Hostname: database-1-admin-mysql.czus806iszm.us-east-2.rds.amazonaws.com

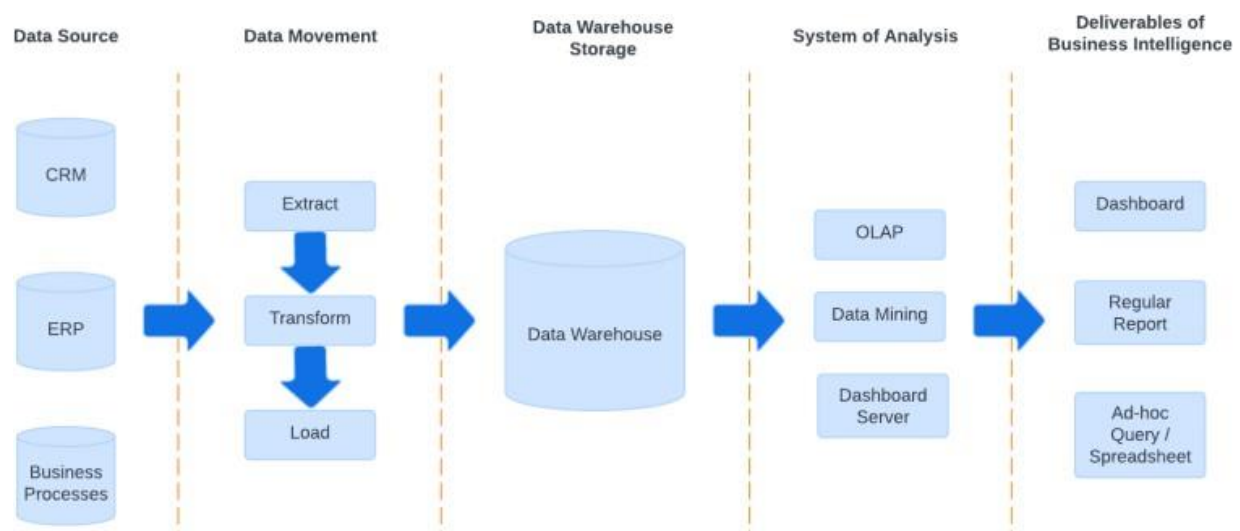
Sunshine Management, LLC

Overview

Sunshine Management, LLC (Sunshine), headquartered in Washington, is dedicated to enhancing the hospitality experience by empowering privately owned properties to meet clients' unique needs and long-term objectives. However, the current system for managing orders, properties, employees, and customer service has faced criticism for its limitations and inability to provide tailored solutions to property owners and customers.

To tackle these challenges head-on, our Business Intelligence (BI) initiative prioritizes user feedback and requirements to develop an integrated cloud-based database. This solution promises swift deployment and scalability, with a projected 38% reduction in service costs and a 40% faster response time for customer requests. By emphasizing user experience and optimizing every stage of the customer journey, from booking to complaint resolution and process updates, we strengthen Sunshine's commitment to flexibility, responsiveness, and customer-centricity.

The accompanying flowchart outlines our approach to database architecture, integrating data from existing systems and centralizing it in a data warehouse. This enables us to create reusable dashboards, reports, and spreadsheets to meet evolving needs, while exploring future enhancements such as data marts for key property owners. This initiative leverages data to serve the diverse requirements of our company, property owners, and customers, fostering innovation and excellence in the hospitality industry.



Business Use Cases of Data Warehouses:

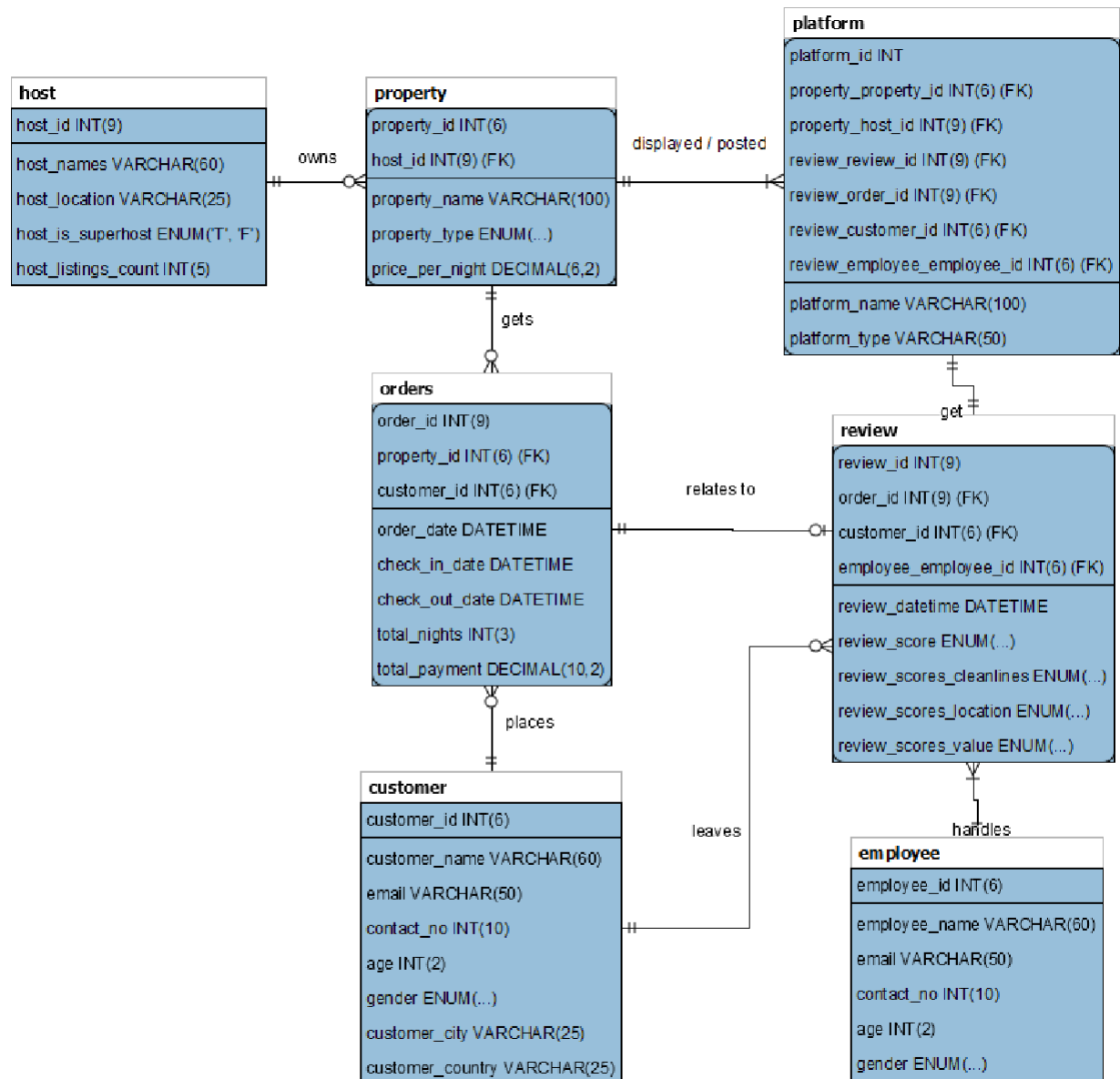
Sunshine embarked on its entrepreneurial journey two decades ago, introducing a groundbreaking business model previously unseen in the hospitality industry. However, with expansion came operational challenges.

- **Data Quality:** The foremost challenge they face is data quality, where manual errors or procedural gaps lead to discrepancies and dissatisfied customers. Additionally, escalating data entry costs, driven by state employee compensation policies, have posed financial strains over the past five years, with a staggering 15% increase. Implementing a centralized database management system addresses these issues by ensuring unified data storage and management, thereby mitigating customer dissatisfaction, and optimizing human resource allocation.
- **Data Integration and System Access:** Furthermore, departmental frustrations at Sunshine stem from difficulties in data integration and system access within the current fragmented systems. Centralization resolves these challenges, fostering collaboration and facilitating information sharing for streamlined decision-making processes.
- **Visibility and Insights on Actionable Items:** Additionally, our commitment to enhancing responsiveness ensures faster resolution of customer requests or complaints, supported by our centralized system offering improved visibility into actionable items. The actionable history at Sunshine is pivotal for analysis and strategic decision-making. Understanding past interactions allows us to pinpoint areas for service enhancement, implementing targeted measures like training programs or system updates to boost customer satisfaction. Leveraging actionable history strengthens customer relationships and loyalty by providing insights into preferences, past purchases, and interactions. This knowledge enables personalized interactions, tailored recommendations, and customized solutions, enhancing engagement and fostering lasting loyalty.
- **Utilization of Available Data Resources:** Moreover, despite possessing a wealth of data, Sunshine has yet to fully capitalize on its potential. By employing processes like ETL (Extract, Transform, Load), data is refined and standardized, enhancing consistency and reliability. The establishment of a new data warehouse enables comprehensive historical analysis, empowering the company to discern trends, patterns, and insights. This initiative not only eliminates the need for external consultants in business intelligence and analytics, saving a substantial amount of expenses, but also position Sunshine as an industry leader. The data warehouse emerges as a strategic asset, enabling Sunshine to harness data for informed decision-making and strategic planning. Furthermore, the envisioned dashboard and knowledge-sharing platform extends benefits to key property owners, fostering mutual growth and setting new industry standards

Database Design

Our BI initiative begins by processing and standardizing data through an ETL process before storing it in our data warehouse. This warehouse comprises seven interconnected tables: host, property, orders, platform, customers, review and employee. These tables exhibit various relationships; some are one-on-one, such as each order corresponding to one review, while

others are one-to-many, such as each customer being able to place multiple orders. The following Entity Relationship (ER) Diagram illustrates the relationships among elements within the data warehouse.



Queries

The physical database shown in the ER Diagram above has been forward engineered into the restaurant's servers.

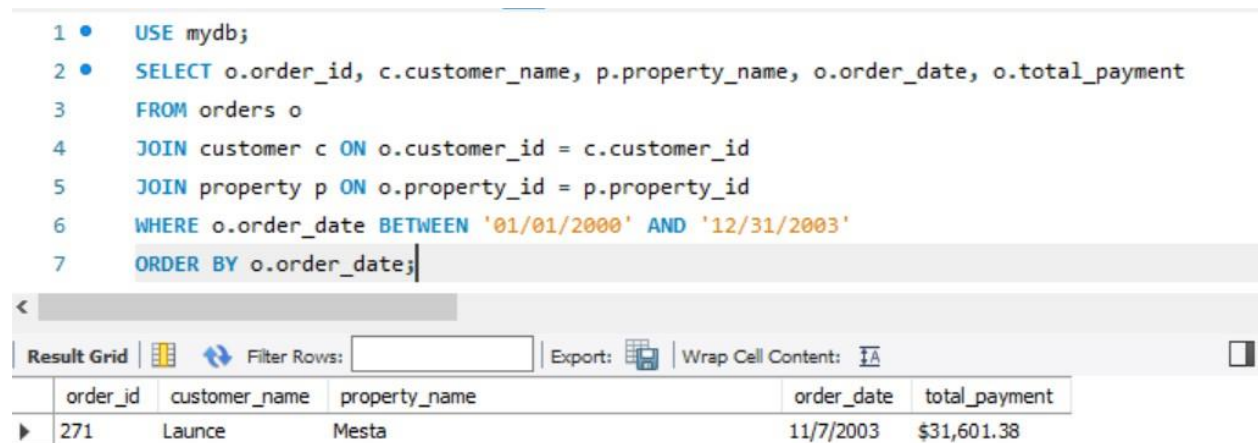
1. To begin, this query fetches all orders placed within a specific date range, which is useful for generating reports for a particular period.

Query for Orders within a Date Range:

USE mm_5075_05;

```
SELECT o.order_id, c.customer_name, p.property_name, o.order_date, o.total_payment
FROM orders o
JOIN customer c ON o.customer_id = c.customer_id
JOIN property p ON o.property_id = p.property_id
WHERE o.order_date BETWEEN '01/01/2000' AND '12/31/2003'
ORDER BY o.order_date;
```

Output:



```
1 • USE mydb;
2 • SELECT o.order_id, c.customer_name, p.property_name, o.order_date, o.total_payment
3 FROM orders o
4 JOIN customer c ON o.customer_id = c.customer_id
5 JOIN property p ON o.property_id = p.property_id
6 WHERE o.order_date BETWEEN '01/01/2000' AND '12/31/2003'
7 ORDER BY o.order_date;
```

order_id	customer_name	property_name	order_date	total_payment
271	Launce	Mesta	11/7/2003	\$31,601.38

2. Relevance of query:

This query identifies customers who have placed repeat orders and their associated review scores and cleanliness ratings. It helps to assess customer loyalty and satisfaction.

Query for Customer Satisfaction:

```
SELECT c.customer_name,
       r.review_score,
       r.review_scores_cleanlines,
       COUNT(o.order_id) AS order_count
FROM customer c
JOIN orders o ON c.customer_id = o.customer_id
JOIN review r ON o.order_id = r.order_id
GROUP BY c.customer_name, r.review_score, r.review_scores_cleanlines
HAVING order_count > 1
ORDER BY order_count DESC;
```

Output:

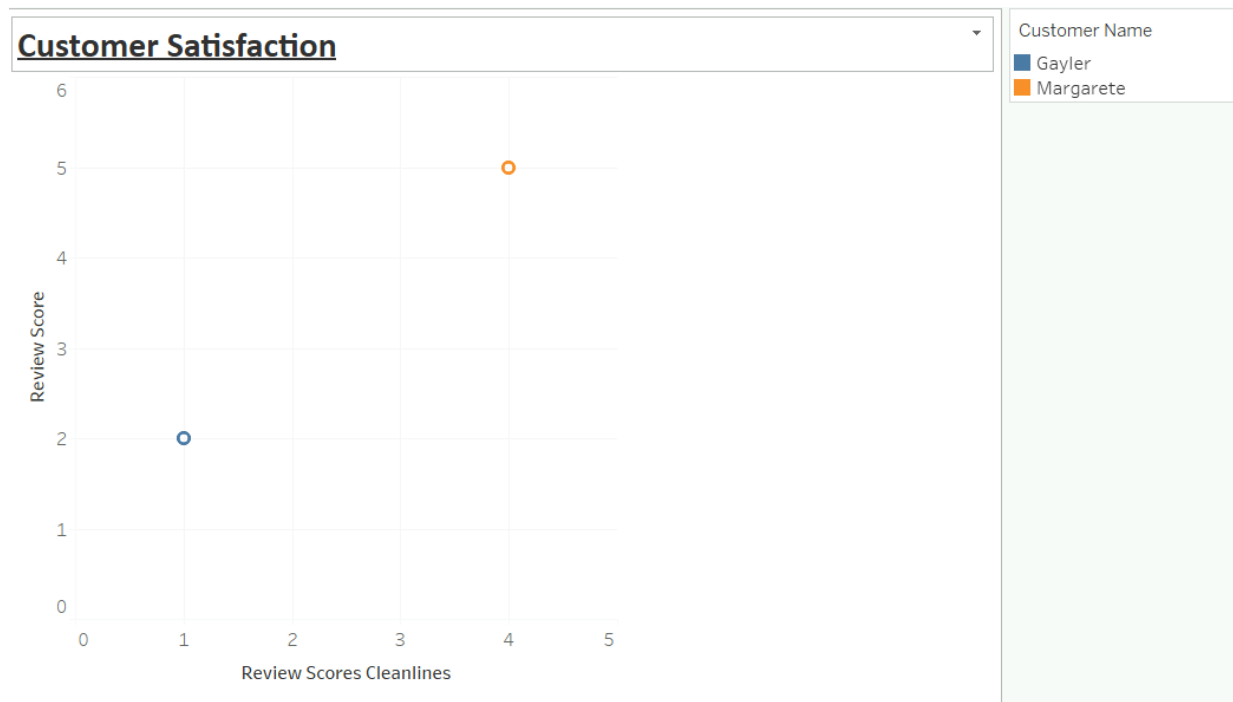
```
2 • SELECT c.customer_name,
3       r.review_score,
4       r.review_scores_cleanlines,
5       COUNT(o.order_id) AS order_count
6 FROM customer c
7 JOIN orders o ON c.customer_id = o.customer_id
8 JOIN review r ON o.order_id = r.order_id
9 GROUP BY c.customer_name, r.review_score, r.review_scores_cleanlines
10 HAVING order_count > 1
11 ORDER BY order_count DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	customer_name	review_score	review_scores_cleanlines	order_count
▶	Gayler	2	1	2
	Margarete	5	4	2

Tableau

<https://public.tableau.com/app/profile/harshitha.m4527/viz/CustomSatisfaction/Sheet1?publish=yes>



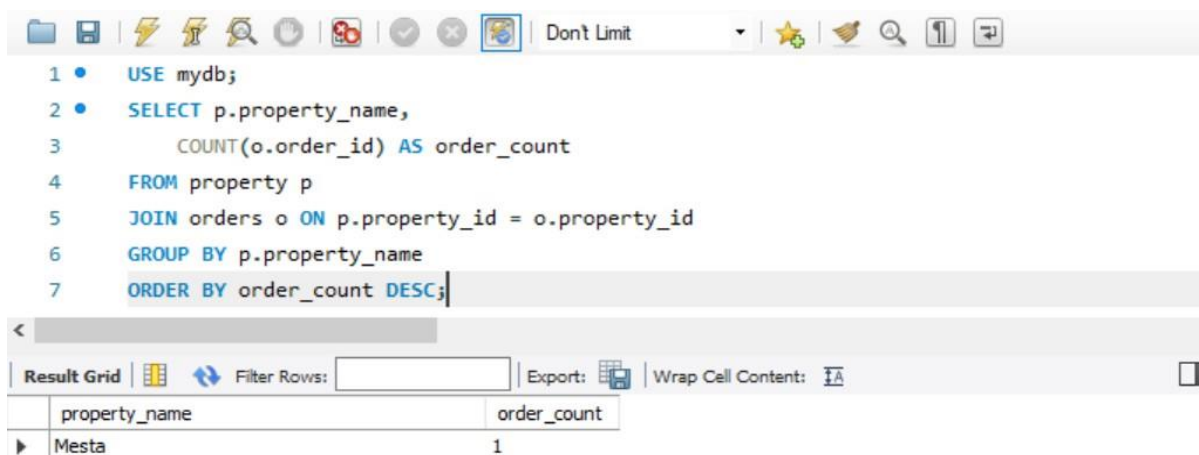
3. Relevance of query:

This query determines which property has the highest number of orders, helping to identify the most popular property and understand customer preferences.

Query for Most Popular Property by Orders:

```
SELECT p.property_name,  
       COUNT(o.order_id) AS order_count  
FROM property p  
JOIN orders o ON p.property_id = o.property_id  
GROUP BY p.property_name  
ORDER BY order_count DESC;
```

Output:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • USE mydb;  
2 • SELECT p.property_name,  
3       COUNT(o.order_id) AS order_count  
4 FROM property p  
5 JOIN orders o ON p.property_id = o.property_id  
6 GROUP BY p.property_name  
7 ORDER BY order_count DESC;
```

Below the query editor is a result grid. The grid has two columns: 'property_name' and 'order_count'. The first row shows 'Mesta' with an order count of 1.

property_name	order_count
Mesta	1

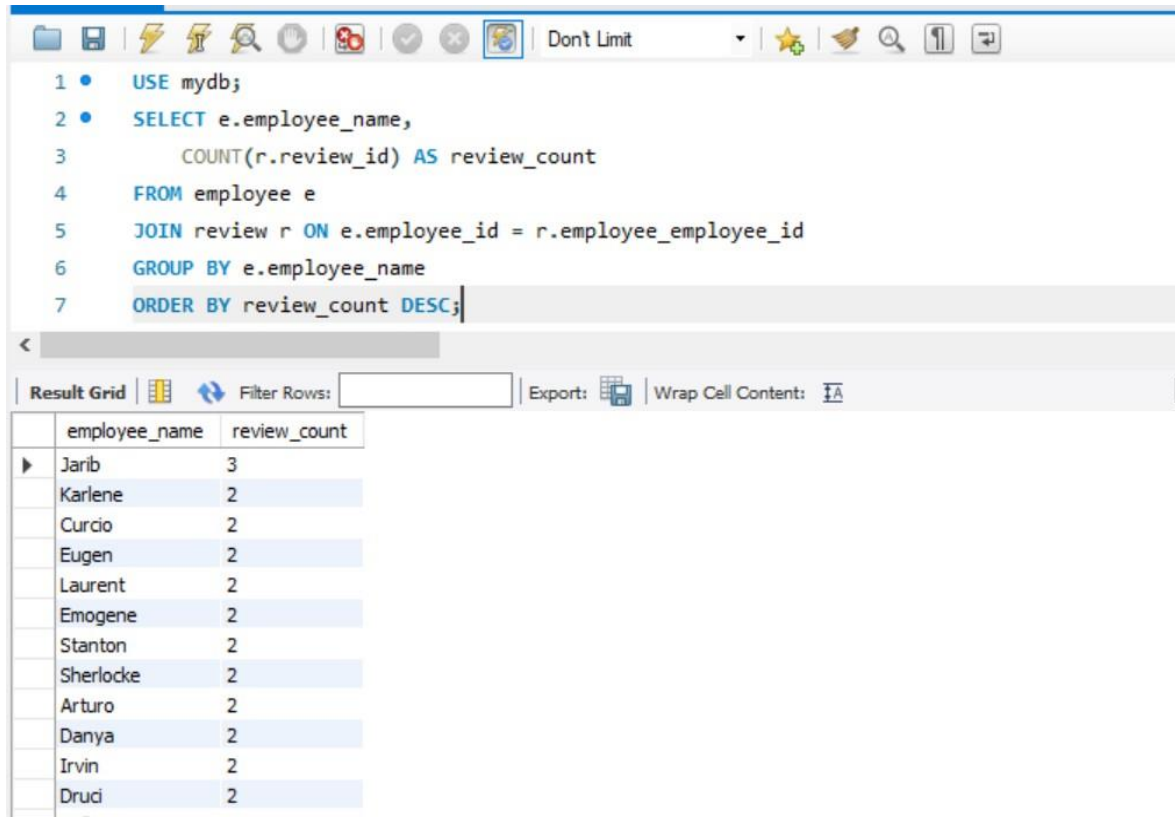
4. Relevance of query:

This query evaluates the performance of employees based on the number of reviews they have handled, which can be used for performance appraisals.

Query for Employee Performance on Review Handling:

```
SELECT e.employee_name,  
       COUNT(r.review_id) AS review_count  
FROM employee e  
JOIN review r ON e.employee_id = r.employee_employee_id  
GROUP BY e.employee_name  
ORDER BY review_count DESC;
```

Output:



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • USE mydb;  
2 • SELECT e.employee_name,  
3       COUNT(r.review_id) AS review_count  
4 FROM employee e  
5 JOIN review r ON e.employee_id = r.employee_employee_id  
6 GROUP BY e.employee_name  
7 ORDER BY review_count DESC;
```

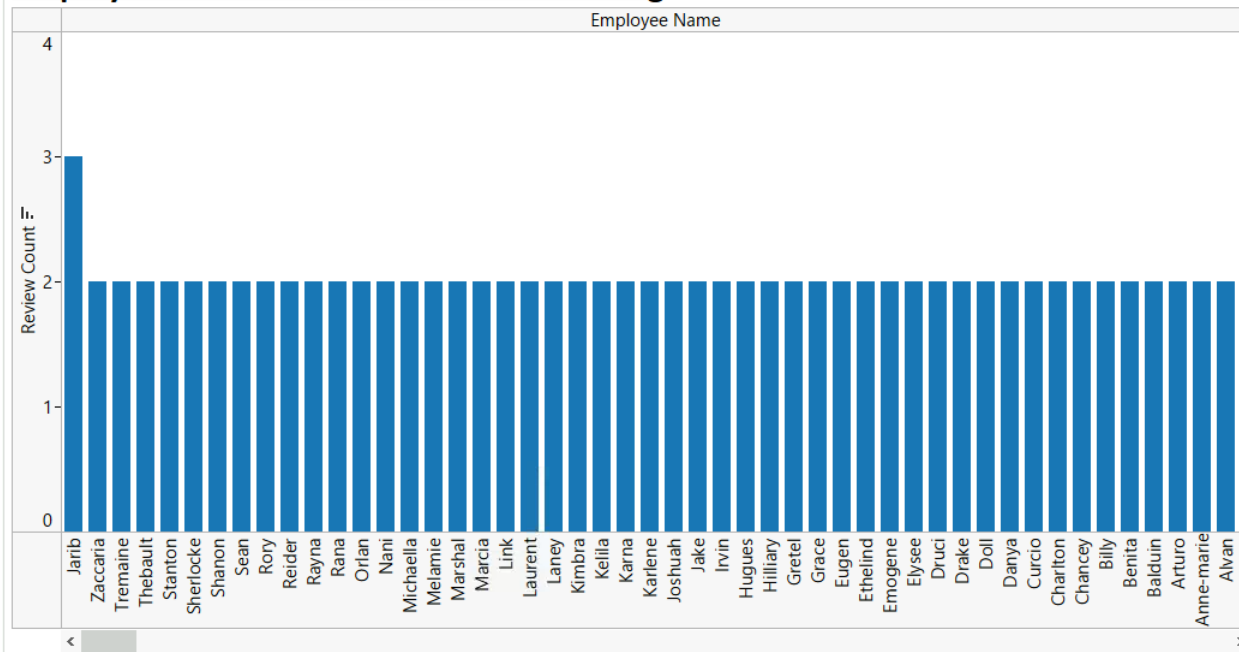
Below the query editor is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The grid displays the following data:

employee_name	review_count
Jarib	3
Karlene	2
Curcio	2
Eugen	2
Laurent	2
Emogene	2
Stanton	2
Sherlocke	2
Arturo	2
Danya	2
Irvin	2
Druci	2

Tableau

https://public.tableau.com/app/profile/harshitha.m4527/viz/Book3_EmployeePerformance/Sheet1?publish=yes

Employee Performance on Review Handling



5. Relevance of query:

This query lists customers along with the number of orders they placed, the number of reviews they wrote, and their average review score. This helps in identifying engaged customers and understanding their satisfaction levels.

Query for Customer Engagement and Satisfaction:

```
SELECT c.customer_name,
       COUNT(o.order_id) AS order_count,
       COUNT(r.review_id) AS review_count,
       AVG(r.review_score) AS avg_review_score
FROM customer c
LEFT JOIN orders o ON c.customer_id = o.customer_id
LEFT JOIN review r ON o.order_id = r.order_id
GROUP BY c.customer_name HAVING order_count > 0
ORDER BY order_count DESC, avg_review_score DESC;
```

Output:

SQL Query:

```

2 • SELECT c.customer_name,
3       COUNT(o.order_id) AS order_count,
4       COUNT(r.review_id) AS review_count,
5       AVG(r.review_score) AS avg_review_score
6 FROM customer c
7 LEFT JOIN orders o ON c.customer_id = o.customer_id
8 LEFT JOIN review r ON o.order_id = r.order_id
9 GROUP BY c.customer_name HAVING order_count > 0
10 ORDER BY order count DESC, avg review score DESC;

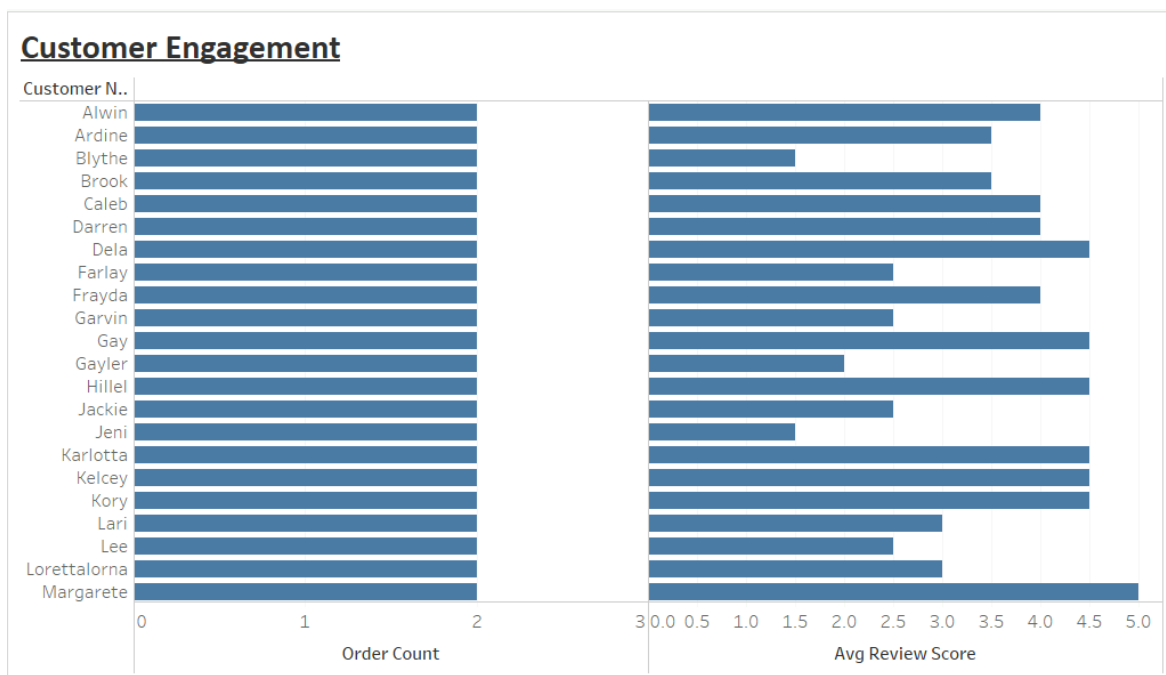
```

Result Grid:

customer_name	order_count	review_count	avg_review_score
Morse	2	2	5.0000
Margarete	2	2	5.0000
Kelcey	2	2	4.5000
Dela	2	2	4.5000
Morgan	2	2	4.5000
Hillel	2	2	4.5000
Gay	2	2	4.5000
Kory	2	2	4.5000
Karlotta	2	2	4.5000
Darren	2	2	4.0000

Tableau

https://public.tableau.com/app/profile/harshitha.m4527/viz/Book5_engagement/Sheet1?publish=yes



https://public.tableau.com/app/profile/harshitha.m4527/viz/Countries_17169557356160/Sheet2?publish=yes



USE of CloverDX to Move to AWS

The old database is being managed using the MySQL Workbench tool. We set up a database in AWS and linked it to Workbench to facilitate the move as much as feasible. The AWS database instance's schema was transferred to Workbench using the schema transfer wizard. The database tables' names were not changed. The ETL pipeline we used to move our data warehouse to the cloud is shown in the graph below. CloverDX, an ETL tool, was utilized to do this task.

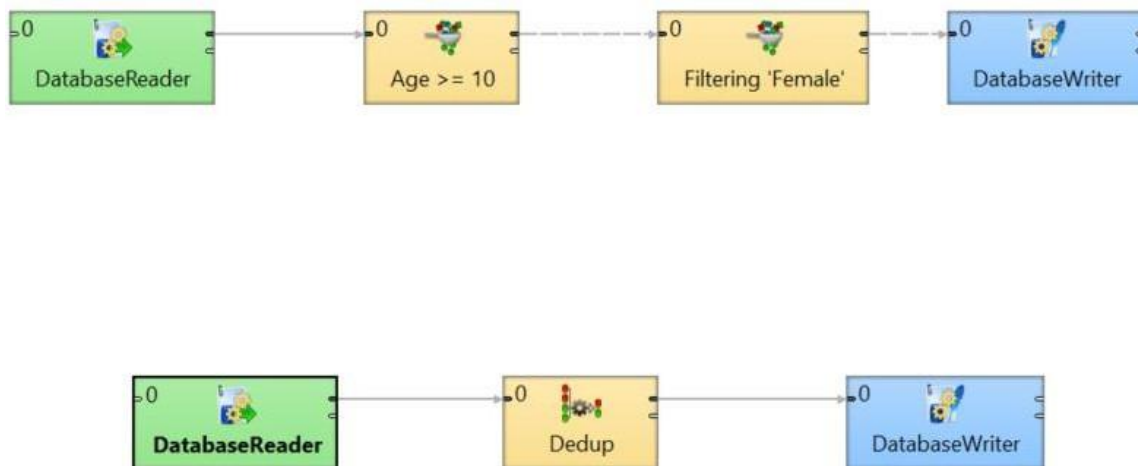
It establishes a connection with both our new AWS servers and our servers, then executes the necessary transfers and transformations for every table. We connected to the new cloud database and the existing database using the appropriate host name, user name, and password in order to use the program (the AWS database credentials are mentioned at the beginning of the article). The pipeline of data transfers and transformations for each table from the source to the cloud database is then depicted in the graph. When the graph is finished, the transfer will be carried out by clicking Run Graph.

Implementation of the CloverDX pipeline:

In CloverDX Designer, create a new project and graph before starting to run a pipeline. Create your task by adding and setting up the required elements, such as DatabaseWriter to write the processed data to a database, Filter to filter data (such as age and gender), and

DatabaseReader to read data. To create the data flow, correctly connect these parts. To guarantee that data structures are handled correctly, define, and configure metadata for every component. The graph can be saved, validated to look for mistakes, and then run by clicking the "Run Graph" button in the toolbar. Keep an eye on the execution in Execution View to see any mistakes or progress. Verify the output in the destination database and check the logs for any problems when the job is finished. If you want the job to run automatically, you might think about deploying it to CloverDX Server. From there, you may schedule the job to run at specific times and configure notifications for its status.

CloverDX pipeline on "employee" table



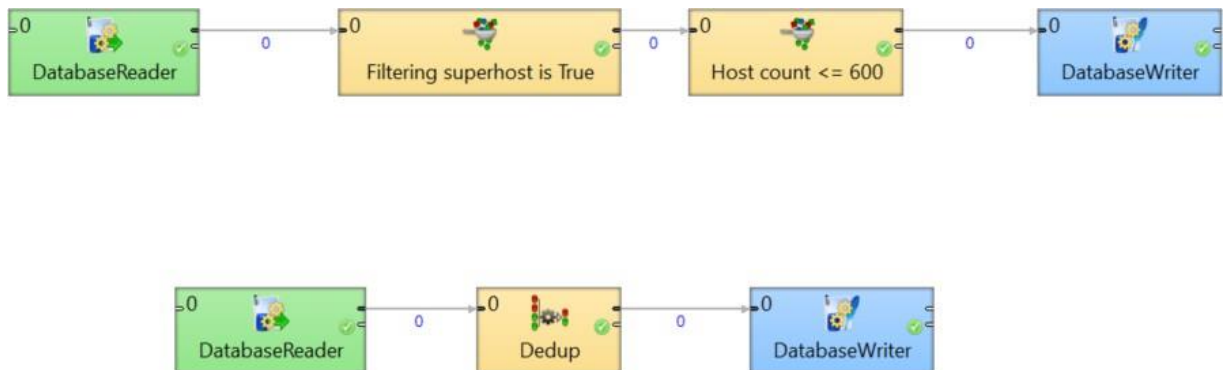
Here in this we will be reading data from a database, specifically fetching records from an employee table.

Transformation Component (Age >= 10): This component filters the records, passing only those where the employee age is greater than or equal to 10.

Transformation Component (Filtering 'Female'): Further filtering is applied, allowing only records of female employees to proceed.

DatabaseWriter: The filtered data is then written to a target RDS database.

CloverDX pipeline on “host” table



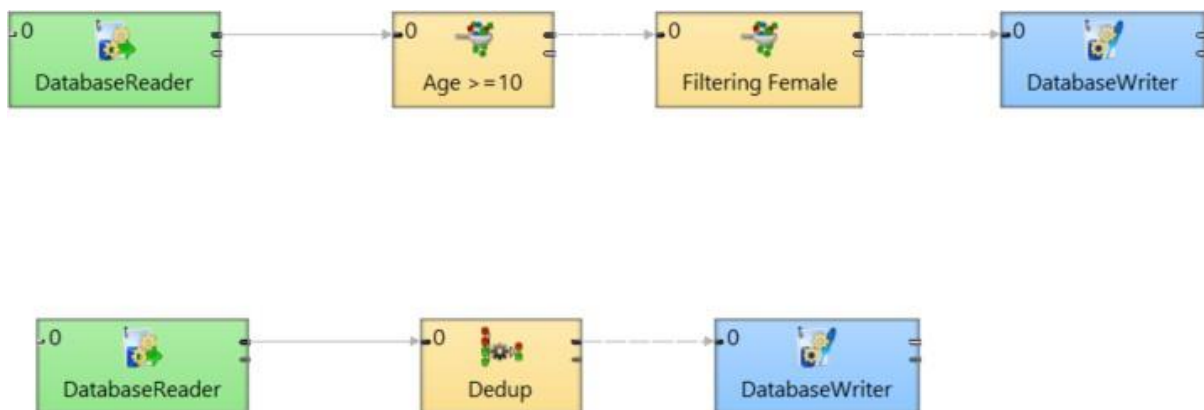
Data is read from a database table, containing information about host table.

Transformation Component (Filtering superhost is True): The data is filtered based on two criteria, one of them is whether a ‘superhost’ flag is set to ‘True’.

Transformation Component (Host count <= 600): The other filtering condition is that the host count should be less than equal 600 counts.

DatabaseWriter: The filtered data is then written to a target RDS database.

CloverDX pipeline on “customer” table



This component reads data(customer) from a database, specifically fetching records from an customer table.

Transformation Component(Age >=10): Data is then filtered by rows where the age is greater than or equal to 10.

Transformation Component(Filtering only “Female”): The data is subsequently filtered again by filtering “Female”.

DatabaseWriter: The filtered data is then written to a target RDS database.

Tableau/ Power BI/Visualization

We made use of Tableau to benefit from the potential of data visualization. Compared to reading and interpreting information from a spreadsheet or straightforward language, humans process visual information more quickly. Seeing all the countries from clients who have placed orders and left evaluations is one of the best examples. As you can see, the site has seen users from all around the nation, with the US, Canada, Laos, and other nations having the largest concentration of visits. It would not have been clear from reading the list of nations and locations connected to the consumers where to look for solutions to the problem with the assistance of this location, but this visualization will assist management in making that decision.

Conclusion

The ETL pipeline for the data transfer from our various data sources was simple to set up thanks to CloverDX. For the sake of ease of transition, we began with the present database schema. We updated the schema to accommodate the growing BI requirements of the company. We can extend the data pipeline in the future to include more automated operations by utilizing CloverDX.

Upcoming modifications to the pipeline will involve establishing a connection with Tableau servers, which will enable real-time data access for the creation of updated visualizations for the business milestones. Updates, additions of new locations, and division of the data into new tables are among the other alterations that can be made. Access to geographical information and customer preferences for those sites will enable the company to adjust to the problem where in which the improvisation needs and, if necessary, introduce seasonal promotions for the betterment of the company. We would be able to monitor platform reviews and obtain more improvised systems with the addition of other tables, such as platforms. These are items to consider when the company expands, but the warehouse we constructed will already offer important new insights to support and take care of its most urgent business requirements.