

AgeX Shipping Company

Capstone Project Report
Data Analysis Using SQL

Prepared by: Aminat Aminu
Date: August 2025

Introduction

This report presents the findings of a data analysis project conducted for AgeX Shipping Company, a global logistics firm operating between the USA, Canada, and Nigeria since February 2020. The company provided shipping and customer data in SQL format, and the goal of this project was to analyze the data and generate insights into customer behavior, sales performance, and shipping trends.

The analysis was carried out using SQL queries in MySQL Workbench. Each query addresses a specific business question, and the results are presented along with explanations of the approach.

Database Design

Below is the ERD (Entity Relationship Diagram) of the database.

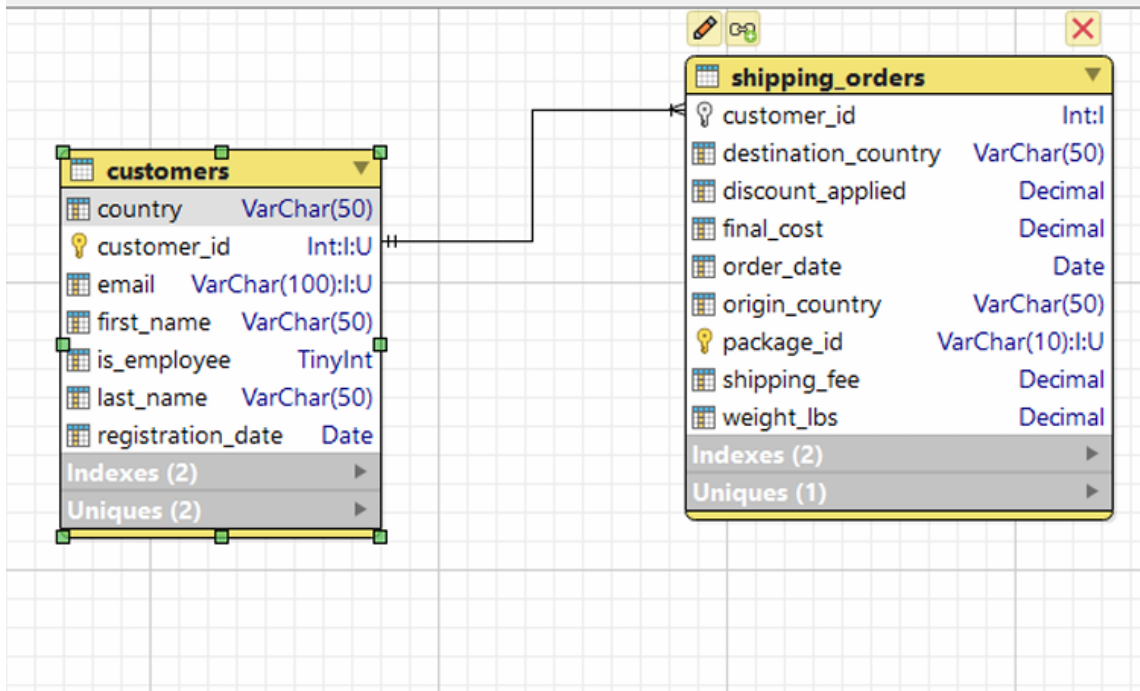


Figure 1: Entity-Relationship Diagram for AgeX Shipping Database

Q1. Total Customers

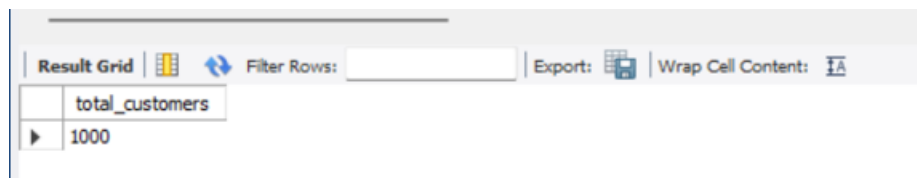
Explanation: I use the `COUNT(customer-id)` to count all registered customers in the database. The query counts all row in the customers table, to return the total number of customers on AgeX database.

SQL Query:

```
--Q1 find the total number of customers registered in AgeX  
-- to count all customer_id values
```

```
SELECT COUNT(customer_id) AS total_customers  
FROM customers;
```

Result:



Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	total_customers			
▶	1000			

Q2. Customers Who Have Shipped

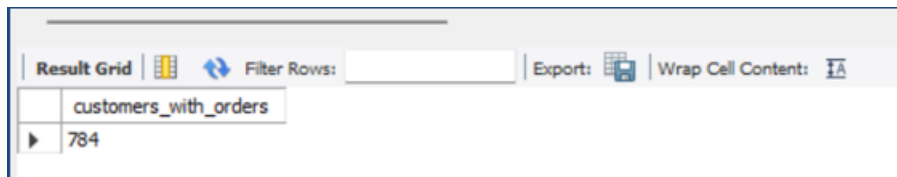
Explanation: I use the `COUNT(DISTINCT)` aggregate function from the shipping table to know how many customers have shipped. Here `DISTINCT` ensures that each customer is counted once even when they appear many times

SQL Query:

```
--Q2 find the number of unique customers who have placed at least one order  
-- count unique customers who have placed at least one order
```

```
SELECT COUNT(DISTINCT customer_id) AS customers_with_orders  
FROM shipping_orders;
```

Result:



The screenshot shows a database interface with a 'Result Grid' tab. The grid has one column labeled 'customers_with_orders' and one row with the value '784'. Above the grid, there are buttons for 'Filter Rows:', 'Export:', and 'Wrap Cell Content:'. The 'Export' button has a small icon next to it.

customers_with_orders
784

Q3. Customers Without Orders

Explanation: To identify customers who have not made any orders but are registered, I use **LEFT JOIN** to connect customers with **shipping_orders**. If a customer has no order, the **s.customer_id** will be NULL, it will filter for those to list the customers without orders.

SQL Query:

```
--Q3 name of customers who have registered but have not placed order  
-- LEFT JOIN customers with shipping_orders  
-- select those where no matching orders exist (NULL)
```

```
SELECT c.first_name, c.last_name  
  
FROM customers as c  
  
LEFT JOIN shipping_orders as s ON c.customer_id=s.customer_id  
  
WHERE c.customer_id is NULL;
```

Result:

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	first_name	last_name			

Q4. Employee Discounts

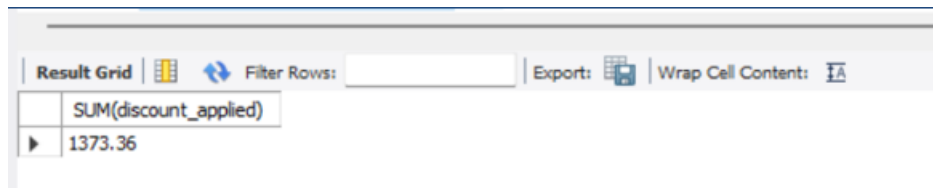
Explanation: I use INNER JOIN to connect shipping_orders with customers because the is_employee column is in the customer table. I WHERE c.is_employee=1 to filter only employee orders. Then I SUM() the discount cost to know how much in total AgeX gave out in discount to employees.

SQL Query:

```
--Q4. total employee discount  
  
-- I SUM up discount_applied for employee (is_employee=1)
```

```
SELECT SUM(discounts_applied)  
FROM shipping_orders as s  
INNER JOIN customers as c ON c.customer_id=s.customer_id  
WHERE is_employee=1;
```

Result:



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains one row with the column header 'SUM(discount_applied)' and a value of '1373.36'. Above the grid, there are controls for 'Filter Rows', 'Export', and 'Wrap Cell Content'.

SUM(discount_applied)
1373.36

Q5. Customers by Region

Explanation: This query counts the number of registered customers in each country. The GROUP BY clause groups the records by country, and COUNT(*) gives the total for each group.

SQL Query:

--Q5. show the total number of customer Located region

-- I use COUNT(*) grouped by country to get totals

```
SELECT country COUNT(*) as total_customers  
FROM customers  
GROUP BY country
```

Result:

Result Grid			Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
	country	total_customers			
▶	Canada	181			
	USA	446			
	Nigeria	373			

Q6. Sales by Region

Explanation: I use SUM(final_cost) to add up all sales from shipping_orders originating in each country. The results are GROUP BY the origin_country.

SQL Query:

```
--Q6. calculate total sales (final_cost) from each origin_country  
-- I use grouped by country to get totals
```

```
SELECT origin_country, SUM(final_cost) as total_sales  
FROM shipping_orders  
GROUP BY origin_country;
```

Result:

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	origin_country	total_sales		
▶	Canada	48704.30		
	Nigeria	83179.32		
	USA	102059.87		

Q7. Sales by Destination

Explanation: This query adds up SUM(final_cost) for each package shipped and delivered to a destination_country, then GROUP BY the destination country

SQL Query:

--Q7 calculate final_cost for packages shipped to each destination_country
-- I use GROUP BY the destination_country

```
SELECT destination_country, SUM(final_cost) as sales_by_destination  
FROM shipping_orders  
GROUP BY destination_country;
```

Result:

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	destination_country	sales_by_destination		
▶	Nigeria	150764.17		
	USA	40184.47		
	Canada	42994.85		

Q8. Sales Over Time

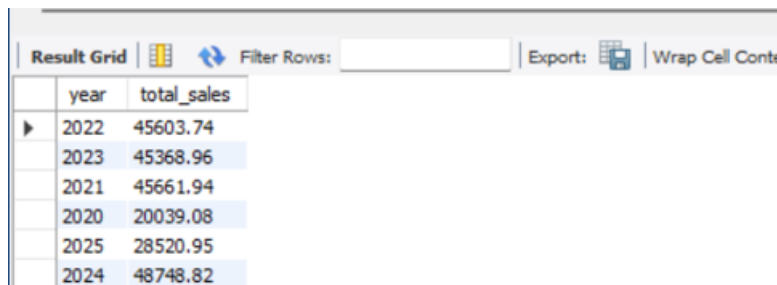
Explanation: I extract the year from order_date, using the YEAR() function, then calculate total sales per year using the SUM(final_cost). The GROUP BY YEAR(order_date) ensures I get the result for total sales for each year.

SQL Query:

```
-- Q8. find total sales for each year since company started  
-- use GROUP BY YEAR() function to extract each year from order_date
```

```
SELECT YEAR(order_date) AS year, SUM(final_cost)AS total_sales  
FROM shipping_orders  
GROUP BY YEAR (order_date);
```

Result:



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of the SQL query, with columns 'year' and 'total_sales'. The data is as follows:

	year	total_sales
▶	2022	45603.74
	2023	45368.96
	2021	45661.94
	2020	20039.08
	2025	28520.95
	2024	48748.82

Q9. Average Shipping Cost per Pound

Explanation: The average shipping cost per pound is found by dividing total shipping cost by the total light_lbs for each origin_country.

SQL Query:

```
-- Q9. find average shipping cost per pound for each origin_country  
-- divide final_cost by light_lb to get the cost per pound
```

```
SELECT origin_country, AVG(final_cost/light_lbs) AS avg_cos_per_lb  
FROM shipping_orders  
GROUP BY origin_country;
```

Result:

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	origin_country	avg_cos_per_lb		
▶	Canada	6.9317697076		
	Nigeria	5.9675716721		
	USA	5.9757569371		

Q10. Most Expensive Order

Explanation: The query sorts all orders by final_cost and packaging_id in descending order and select the top one with the highest value, I used WHERE final_cost IS NOT NULL to remove incomplete or missing cost records.

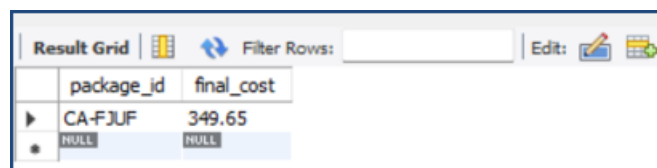
The final query finds the most expensive order by checking the highest value in the final_cost, results show packaging_id along with its maximum cost. In the final result, there is also a NULL value which means that at least one order final_cost do not have a recorded final_cost in the database.

SQL Query:

-- Q10. find the most expensive order (excluding NULL values)

```
SELECT package_id, final_cost
FROM shipping_orders
WHERE final_cost IS NOT NULL
ORDER BY final_cost DESC
LIMIT 1;
```

Result:



	package_id	final_cost
▶	CA-FJUF	349.65
*	NULL	NULL

Q11. High-Value Customers

Explanation: Calculate the total spending per customer, then use the HAVING clause to keep only those above \$500. Finally, ORDER BY from highest to lowest spender.

SQL Query:

-- Q11. find the customers who have spent more than \$500 in total
-- I SUM the final_cost per customer then use the HAVING clause to filter

```
SELECT customer_id, SUM(final_cost) AS total_spent
FROM shipping_orders
GROUP BY customer_id
HAVING SUM(final_cost) > 500
ORDER BY total_spent DESC;
```

Result:

Result Grid			Filter Rows:	Export:	Wrap Cell C
	customer_id	total_spent			
▶	302	1110.84			
	729	1077.37			
	960	1035.66			
	157	1029.07			
	461	957.53			
	696	948.60			
	392	940.80			
	821	931.32			
	386	892.71			

Q12. Sales from Specific Region (Nigeria to Canada)

Explanation: I filter shipment by origin (Nigeria) and destination (Canada) and then sum of their final_cost to get the total revenue.

SQL Query:

```
-- Q12. total revenue from shipments originating in Nigeria destined Canada  
-- apply WHERE filter for both origin and destination
```

```
SELECT SUM(final_cost) AS total_revenue  
FROM shipping_orders  
WHERE origin_country='Nigeria'  
AND Destination_country='Canada';
```

Result:

Result Grid		Filter Rows:	Exp
	total_revenue		
▶	42994.85		

Q13. Employee's Sales (Orders > 30 lbs)

Explanation: I JOIN customers with shipping orders to identify employee shipments.
And is_employee=1 only count those where light_lbs is equal to 30lbs

SQL Query:

```
-- Q13. find total sales generated by employee shipment  
-- JOIN customers (to check is_employee) with shipping_orders  
-- only include shipments greater than 30lbs
```

```
SELECT SUM(s.final_cost) AS total_employee_sales  
FROM shipping_orders as s  
INNER JOIN customers as c  
ON c.customer_id=s.customer_id  
WHERE c.is_employee=1  
AND s.light_lbs > 30;
```

Result:

Result Grid		Filter Rows:	Export:	Wrap Cell C
total_employee_sales				
▶	7267.43			

Q14. Popular Routes

Explanation: I group shipments by routes (Origin → Destination), count the number of packages, sort them in descending order and take the top 3.

SQL Query:

```
-- Q13. find the top 3 most popular shipping routes (from origin_country to destination_country)

--GROUP BY origin_country and destination_country

-- ORDER BY package_count
```

```
SELECT origin_country, destination_country, COUNT(*) As package_count
FROM shipping_orders
GROUP BY origin_country, destination_country
ORDER BY package_count DESC
LIMIT 3;
```

Result:

Result Grid	Filter Rows:	Export:	Wrap Cell C
origin_country	destination_country	package_count	
USA	Nigeria	668	
Nigeria	Canada	284	
Canada	Nigeria	277	

Q15. Customers with Last Name Starting with 'O'

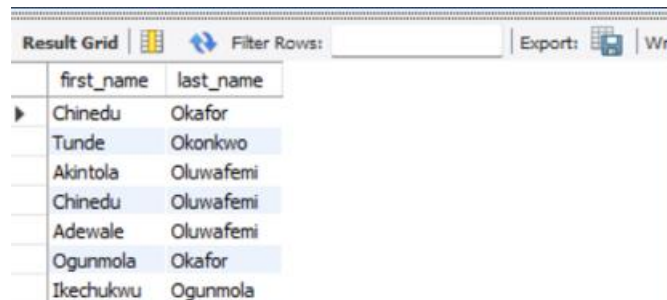
Explanation: I use LIKE 'O%' conditions to filter customers whose last name begins with "O"

SQL Query:

```
-- Q15. find all customers whose last_name LIKE 'O'
-- use LIKE with a wildcard
```

```
SELECT first_name, last_name
FROM customers
WHERE last_name LIKE 'O%';
```

Result:



The screenshot shows a 'Result Grid' window with a table of customer data. The table has two columns: 'first_name' and 'last_name'. The data is as follows:

first_name	last_name
Chinedu	Okafor
Tunde	Okonkwo
Akintola	Oluwafemi
Chinedu	Oluwafemi
Adewale	Oluwafemi
Ogunmola	Okafor
Ikechukwu	Ogunmola

Q16. Package Tracking String

Explanation: I use CONCAT to merge the package ID, origin country and destination country into a readable sentence for tracking.

SQL Query:

```
-- Q16. create a readable tracking strings  
  
-- use CONCAT to combine packaging_id, origin_country, destination_country
```

```
SELECT CONCAT('package', ' ', package_id, ' ', 'from', ' ', origin_country, ' ', 'to', ' ',  
' ', destination_country) AS tracking_info  
  
FROM shipping_orders;
```

Result:

Result Grid		Filter Rows:	Export:
	tracking_info		
▶	package CA-04I8 from Canada to Nigeria		
	package CA-067W from Canada to Nigeria		
	package CA-071Q from Canada to Nigeria		
	package CA-0C5O from Canada to Nigeria		
	package CA-0FLO from Canada to Nigeria		
	package CA-0LQG from Canada to Nigeria		
	package CA-00OQ from Canada to Nigeria		
	package CA-0PGW from Canada to Nigeria		
	package CA-0RPM from Canada to Nigeria		

Q17. Top Spenders by country 'USA'

Explanation: I JOIN customers and orders, filter for costumers in the USA, calculate their total spending and list the top 5 spenders ordered by total spent.

SQL Query:

```
-- Q16. create a readable tracking strings  
-- use CONCAT to combine packaging_id, origin_country, destination_country
```

```
SELECT CONCAT(c.first_name,' ',c.last_name,' ',c. country) AS full_name,  
SUM(s.final_cost) AS total_spent  
  
FROM customers as c  
  
JOIN shipping_orders as s ON c.customer_id=s.customer_id  
  
WHERE c.country= 'usa'  
  
GROUP BY c.customer_id, first_name, last_name  
  
ORDER BY total_spent DESC LIMIT 5;
```

Result:

Result Grid			Filter Rows:	Export:	Wrap
	full_name	total_spent			
▶	Michael David USA	1035.66			
	Robert Emily USA	931.32			
	Michael Chris USA	861.90			
	William John USA	853.08			
	Robert Ashley USA	842.88			

Conclusion

This analysis provided insights into:

- Customer demographics
- Revenue trends
- Shipping patterns
- Employee discounts impact

The findings can help AgeX Shipping improve decision-making and optimize operations.