Introduction

This project focuses on creating a data warehouse for the Sakila database. The sakila database is a popular MySQL database showing a business film rental activity. It records the movies being rented out, the payments for each rental, the customer details, and many other records. See more about it here.

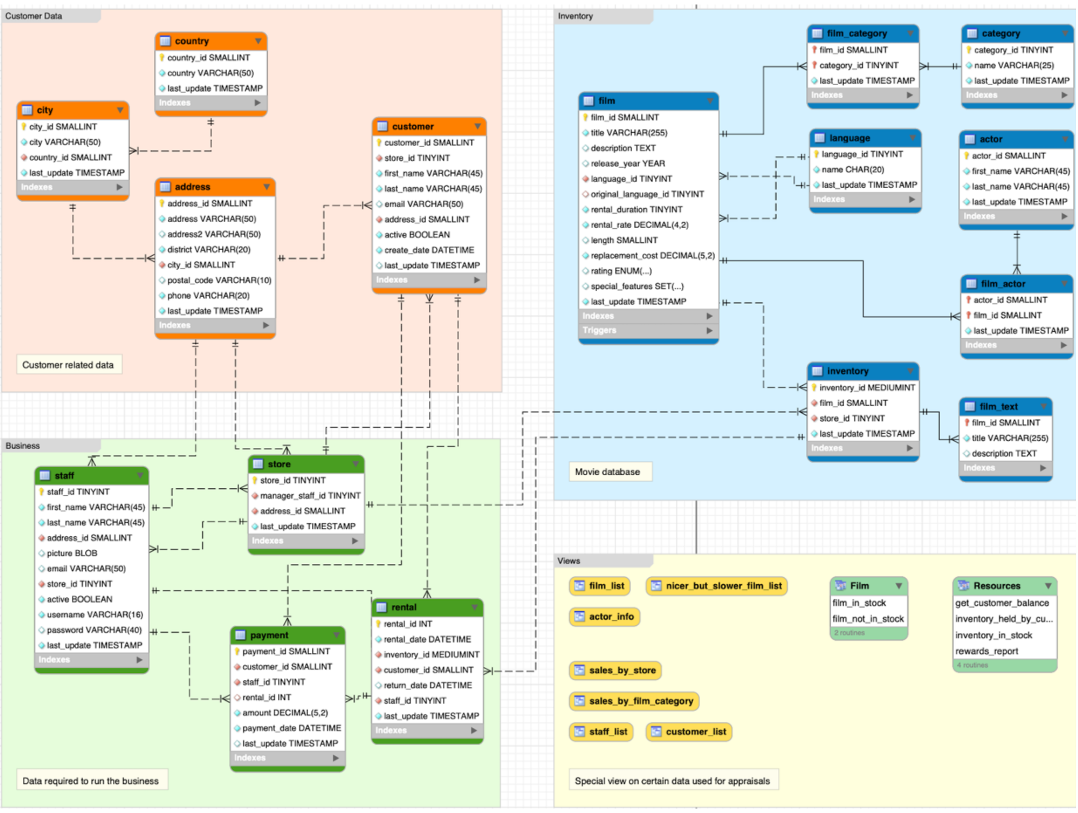


Figure 1: Sakila database schema

Several analyses can be drawn from this database, including sales-based analysis, order-based analysis, customer-based analysis, etc. For this project we have decided to pay attention on the inventory analysis.

For a rental business it is very paramount to the business owners to know at all point what current stock of the movies are available at every given time, to also monitor how the movies are moving, where they are at certain times. Hence, we will be creating a data warehouse that helps the inventory manager keep tabs of all movies in stock for every store of the business.

Building the Data Warehouse

The goal is to develop a data warehouse that integrates various dimensions like customers, films, rentals, calendar, and store to have a unified view of the business operations. This integration will allow for efficient inventory management, enhanced customer experience, and optimized rental processes.

The data warehouse answers one most important question amongst others; How can we efficiently monitor and manage movie inventory across all store locations to ensure optimal availability, and tracking of movie stock? To achieve this, we have built a star schema that includes five (5) dimension tables and an inventory fact table that records the total inventory quantity and the movie availability status. Here is what the schema look like.

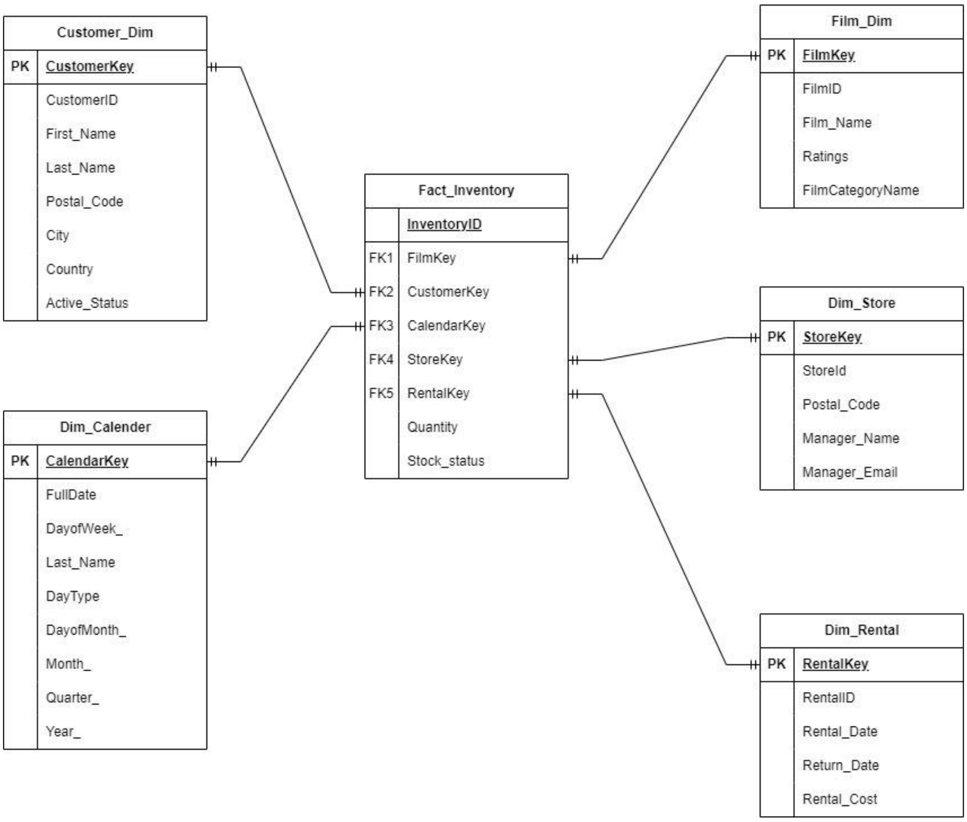


Figure 2: Sakila Inventory datawarehouse

This data warehouse answers what we call the 3Ws; what movies are available, when are they (or will they be) available, and lastly who has the rented movies. With the selected details we can identify what quantity of movies the business has across all stores, which of them are available and which isn't, who has the rented movies, and so on.

Purpose of the Tables

The **film dimension** table allows us capture what films we have in the database, the total quantities, and their availability status. The **rental dimension** table allows us capture what movies are currently rented out, the cost for rentals, the return date for every rental, etc. and so on. The **calendar dimension table** helps us monitor the dates rental activities in the database. With this we can tell the total inventory available at every given date. The **customer table** helps us monitor what customers have sets of the films across the globe, this makes it easy to track each movie rented out. The **store dimension table** helps us monitor the inventory in each store at every given time.

Business Objectives

From the data we seek to understand the following insights

1. Determine the current inventory levels for each film in the database to facilitate efficient stock management.
2. Calculate the total inventory across various store locations to optimize distribution and ensure sufficient availability.
3. Analyze the geographic distribution of films to identify market trends and tailor inventory strategies accordingly.
4. Identify movies with the highest quantities in the database to understand popular demand and inform purchasing decisions.
5. Monitor stock levels to identify films at risk of running low and take proactive measures to replenish inventory.

The key stakeholders will find these insights helpful are:

1. **Business Analysts**: They analyze the data for insights into customer preferences and behavior.
2. **Management Team**: They utilize the insights derived from the data to make strategic decisions.
3. **Marketing Team:** Insights into customer demographics and movie preferences can inform targeted marketing campaigns to promote specific movies to likely renters.
4. **Store Managers:** Data from the warehouse can empower store managers to optimize inventory levels at their specific stores, identify popular genres among their customer base, and tailor promotions accordingly.

Designing the ETL

In this session we focus on extracting data from the database, manipulating them to fit the records we want to store, and loading them into the data warehouse. We achieve this process using the SQL Server Integration Services (SSIS)tool. The ETL follows a simple plan; extract data from the database, transform it and load into our database. Here is a visual representation of the idea.

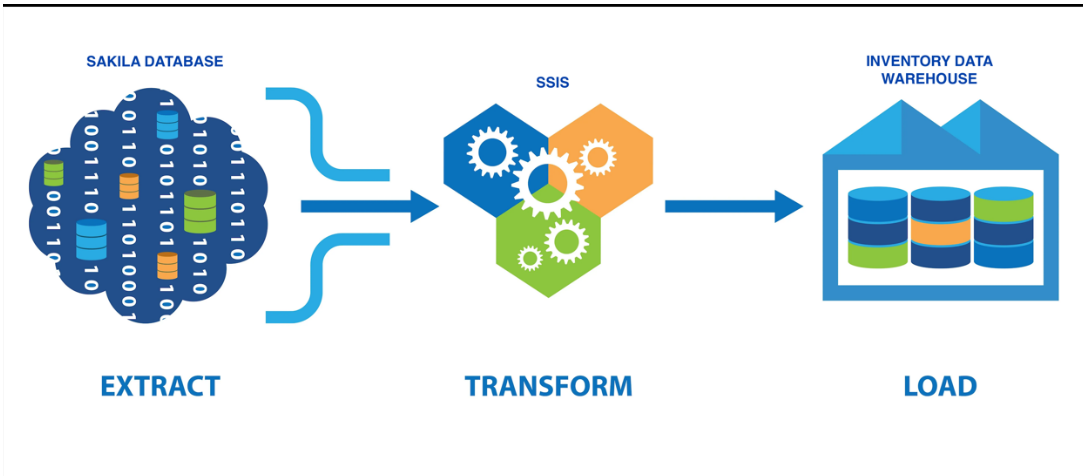


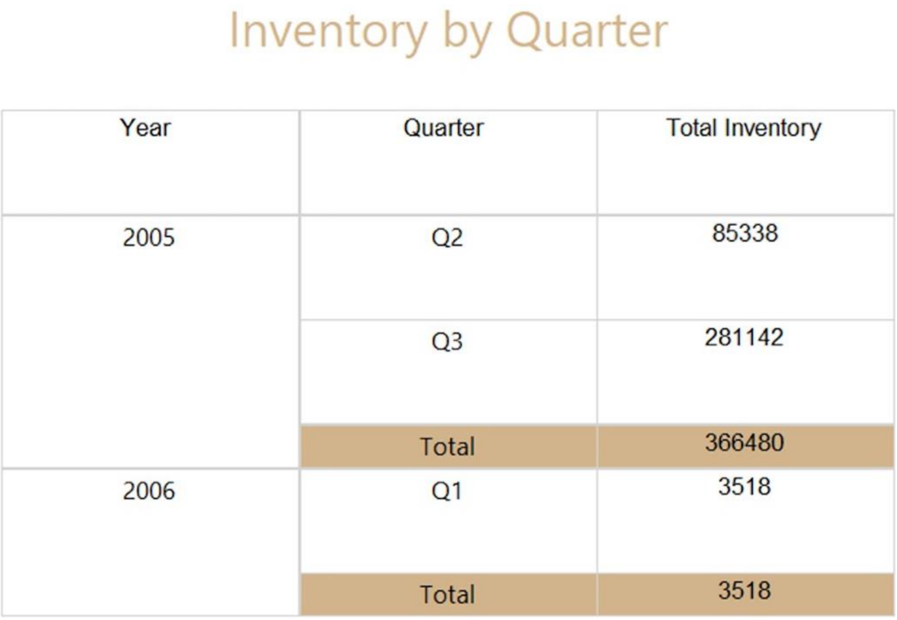
Figure 3: ETL process

Analytical Report

In this analysis, we seek to help stakeholders achieve certain insights from the data warehouse using SQL Server Reporting Services (SSRS) and Tableau. We have built visuals from both platforms to help give better understanding of the dataset.

Quarter Inventory for each Year:

This report provides an analysis of the inventory levels for a company for the years 2005 and 2006, broken down by quarter. The data in this report can help the company make informed decisions about inventory management and identify any trends or fluctuations in inventory levels.



Inventory by Quarter

Year	Quarter	Total Inventory
2005	Q2	85338
	Q3	281142
	Total	366480
2006	Q1	3518
	Total	3518

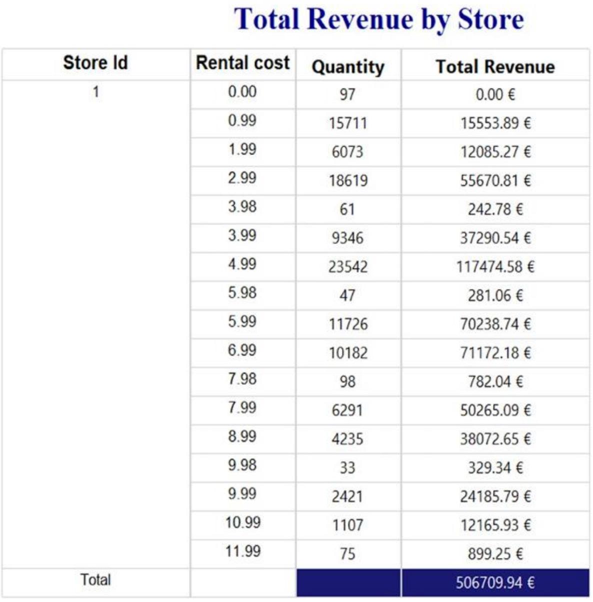
Figure 4: Inventory by

The table titled "Inventory by Quarter" shows the total inventory for each quarter in 2005 and 2006. In 2005, the inventory levels started at 85,338 in Q2 and increased significantly to 281,142 in Q3. The total inventory for 2005, combining Q2 and Q3, was 366,480.Moving on to 2006, the inventory levels decreased drastically to 3,518 in Q1.

Total Revenue by Store:

This report provides insight into the total revenue made by each store across the years in terms of rental cost and quantity rented. This data gives relevant stakeholders an idea on how the stores are performing.

From this report we can see the overall revenue recorded by store1, and we are also able to filter the table by the different stores to get a view of the total revenue.



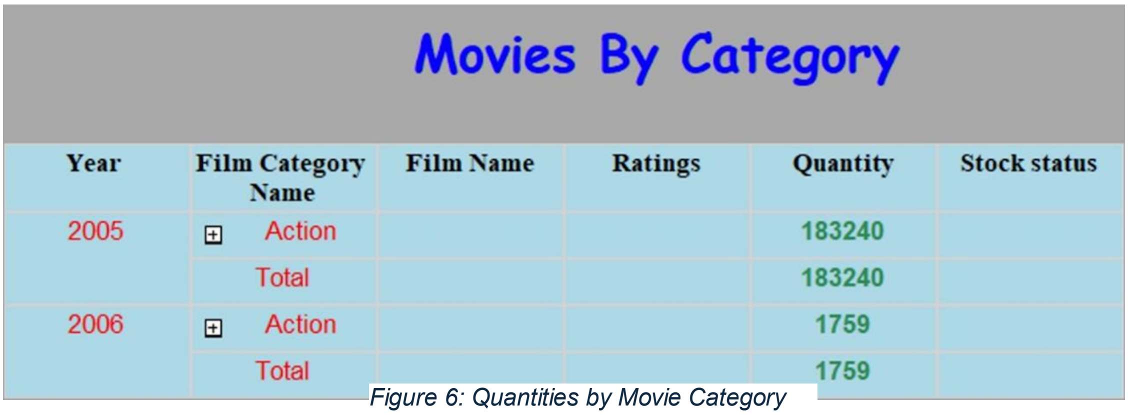
Total Revenue by Store

Store Id	Rental cost	Quantity	Total Revenue
1	0.00	97	0.00 €
	0.99	15711	15553.89 €
	1.99	6073	12085.27 €
	2.99	18619	55670.81 €
	3.98	61	242.78 €
	3.99	9346	37290.54 €
	4.99	23542	117474.58 €
	5.98	47	281.06 €
	5.99	11726	70238.74 €
	6.99	10182	71172.18 €
	7.98	98	782.04 €
	7.99	6291	50265.09 €
	8.99	4235	38072.65 €
	9.98	33	329.34 €
	9.99	2421	24185.79 €
	10.99	1107	12165.93 €
	11.99	75	899.25 €
Total			506709.94 €

Figure 5: Revenue by

Movies by Category:

This report produces a drill down table to allow stakeholders view various movies in each movie categories and get a hang of their ratings, the total quantity, and their availability status.



Movies By Category

Year	Film Category Name	Film Name	Ratings	Quantity	Stock status
2005	⊞ Action			183240	
	Total			183240	
2006	⊞ Action			1759	
	Total			1759	

Figure 6: Quantities by Movie Category

Movies with the Most Quantity:

This analysis allows stakeholders identify movies with the most quantities in the database. With this, it is easy to tell what movie are being prioritized and sorted after.

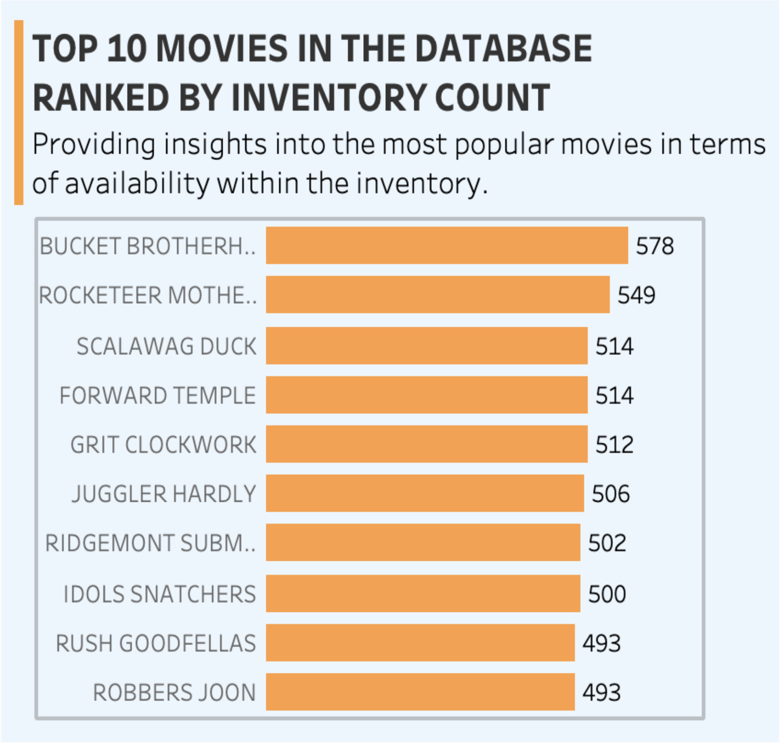


Figure 7: Movies with the most quantities

From this we can see that the Bucket Brothers and the Rocketeer mothers have the most inventory quantities in the database.

Most Expensive Movies:

This analysis enables stakeholders to identify movies with high rental cost. With this the business takes note of movies that provides the most revenue and prioritize them.

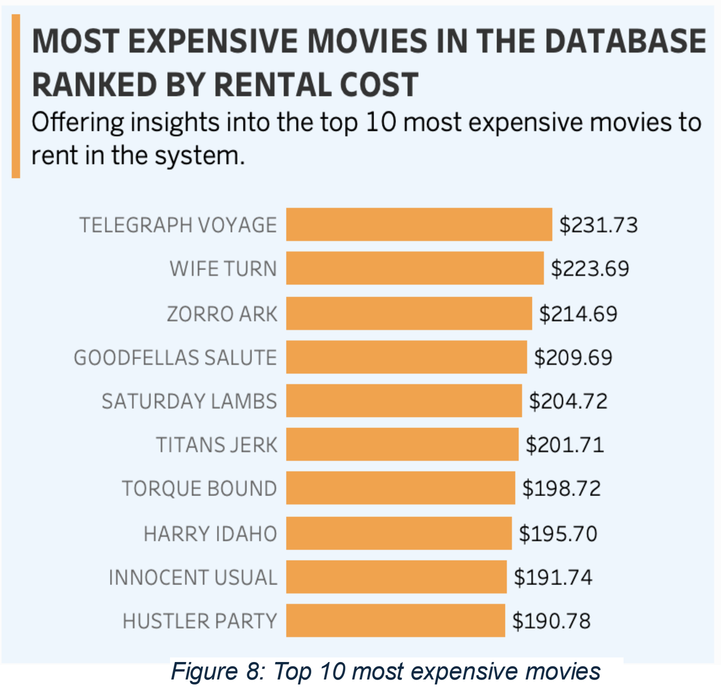


Figure 8: Top 10 most expensive movies

From this table, we can see the Telegraph Voyage and Wife Turn movies have the higher rental cost and thus potentiality to generate more revenue.

Inventory Distribution Across Stores:
This analysis enables stakeholders keep stock of the total number of inventories across the business stores.

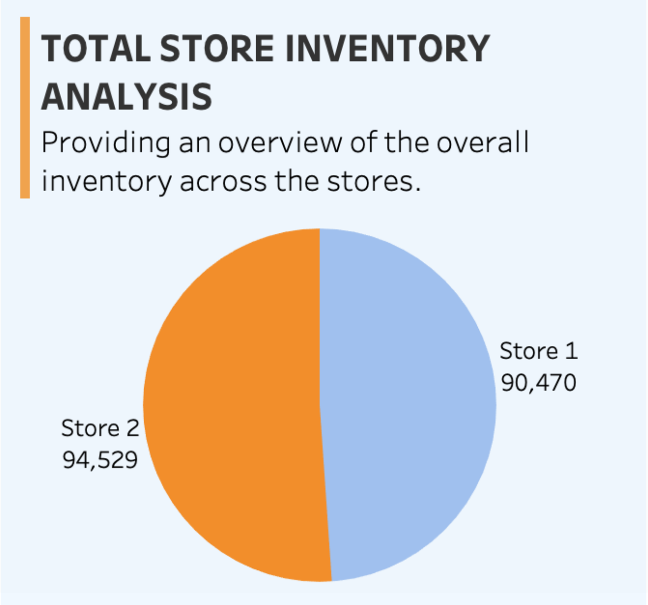


Figure 9 Inventory Per Store

Global Inventory Distribution:

This analysis enables stakeholders to monitor the spread of the films across the globe. With this, managers can tell where each movies rented are currently located, and what quantity of the movies can be found in these locations.

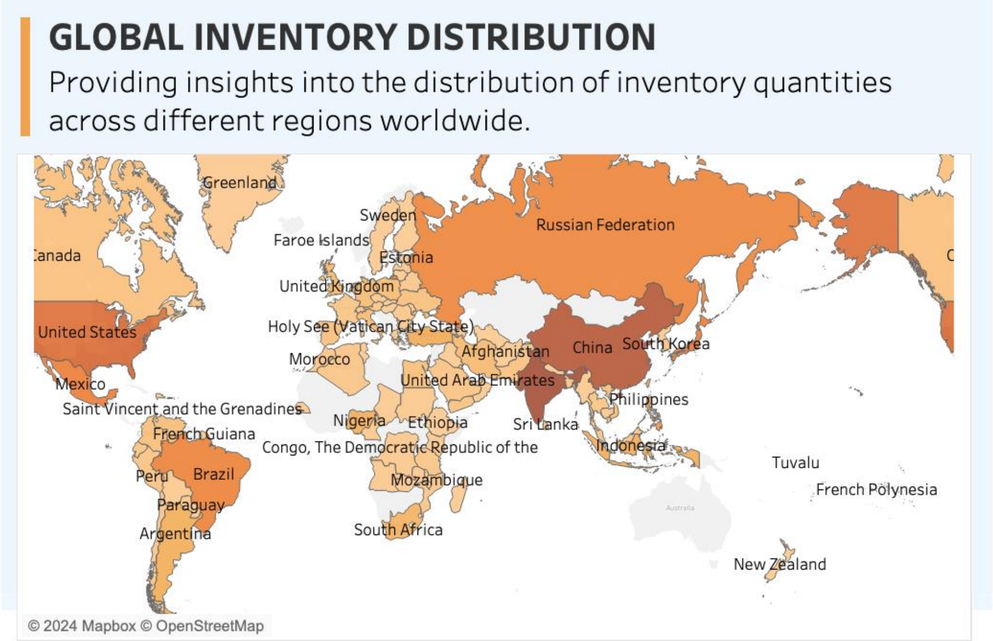


Figure 10: Distribution of Movies across the globe

SQL VS Graph Databases

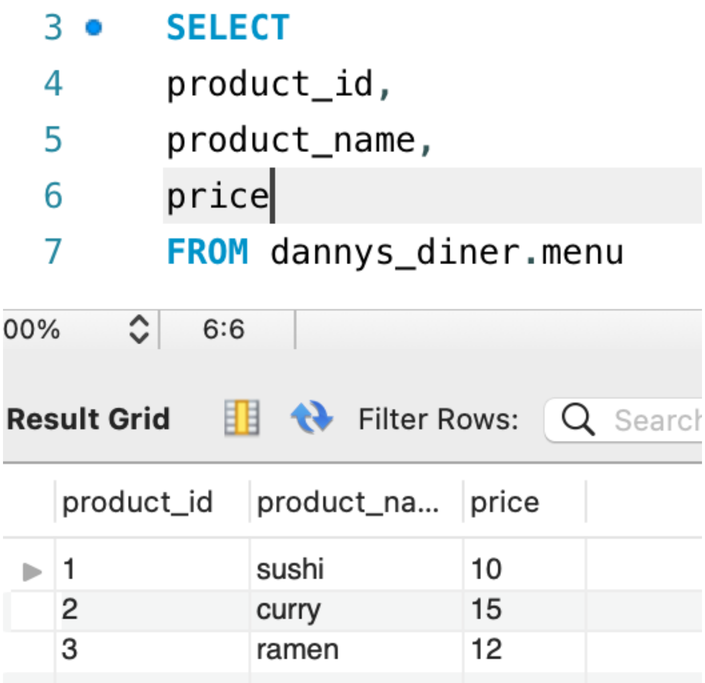
SQL (Structured Query Language) and graph databases are two different approaches to managing and querying data in a database.

SQL databases, also known as relational databases, store data in tables, with each table consisting of rows and columns. Each row in a table represents a record, and each column represents a field in the record. Graph databases, on the other hand, are non-relational databases that store data in a graph-like structure consisting of nodes and edges. Nodes represent entities, and edges represent the relationships between them.

In this section, we will compare seven various queries of both SQL and graph database to get an understanding of how both works. The dataset we will be using is the Danny's Diner dataset. Find it here.

Question 1: Write a simple select statement to show the details on the Menu Table

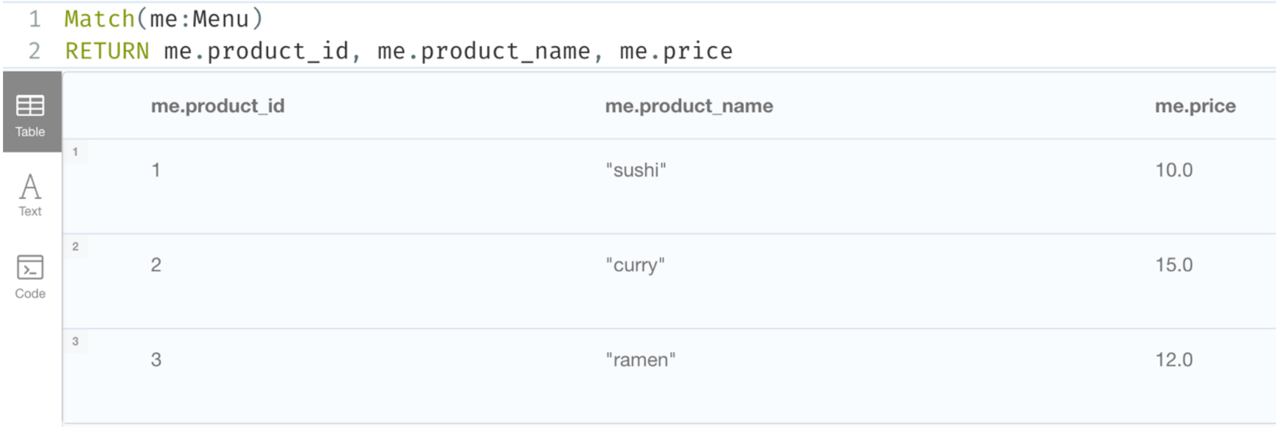
SQL→



```
SELECT
product_id,
product_name,
price
FROM dannys_diner.menu
```

product_id	product_na...	price
1	sushi	10
2	curry	15
3	ramen	12

Cypher Query →

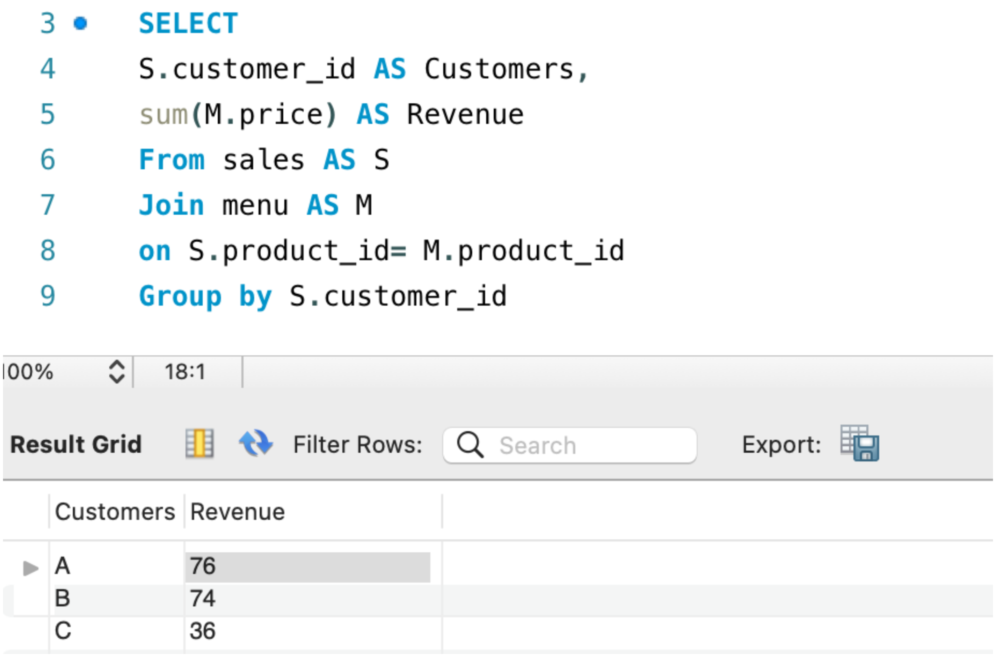


```
Match(me:Menu)
RETURN me.product_id, me.product_name, me.price
```

me.product_id	me.product_name	me.price
1	"sushi"	10.0
2	"curry"	15.0
3	"ramen"	12.0

Question 2: Write a query to show the total revenue generated by each customer.

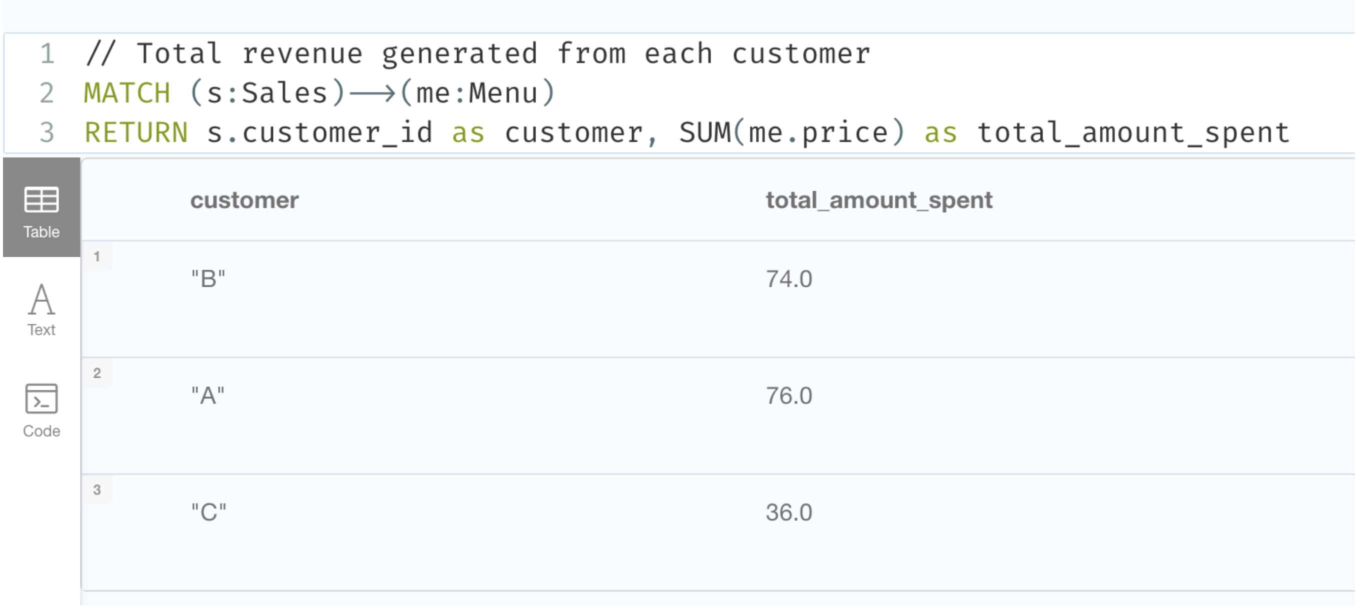
SQL →



```sql
SELECT
S.customer_id AS Customers,
sum(M.price) AS Revenue
From sales AS S
Join menu AS M
on S.product_id= M.product_id
Group by S.customer_id
```

Customers	Revenue
A	76
B	74
C	36

Cypher Query →

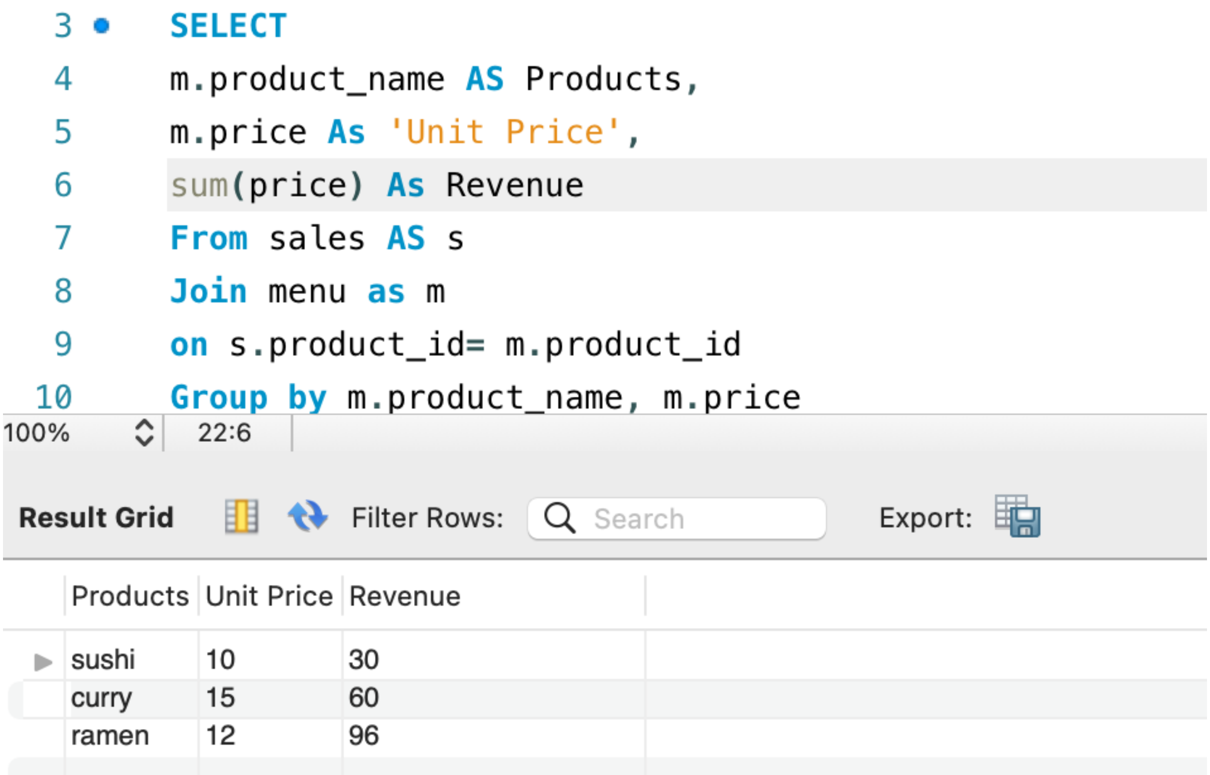


```
// Total revenue generated from each customer
MATCH (s:Sales)⟶(me:Menu)
RETURN s.customer_id as customer, SUM(me.price) as total_amount_spent
```

	customer	total_amount_spent
1	"B"	74.0
2	"A"	76.0
3	"C"	36.0

Question 3: Write a query to show the total revenue generated by each product.

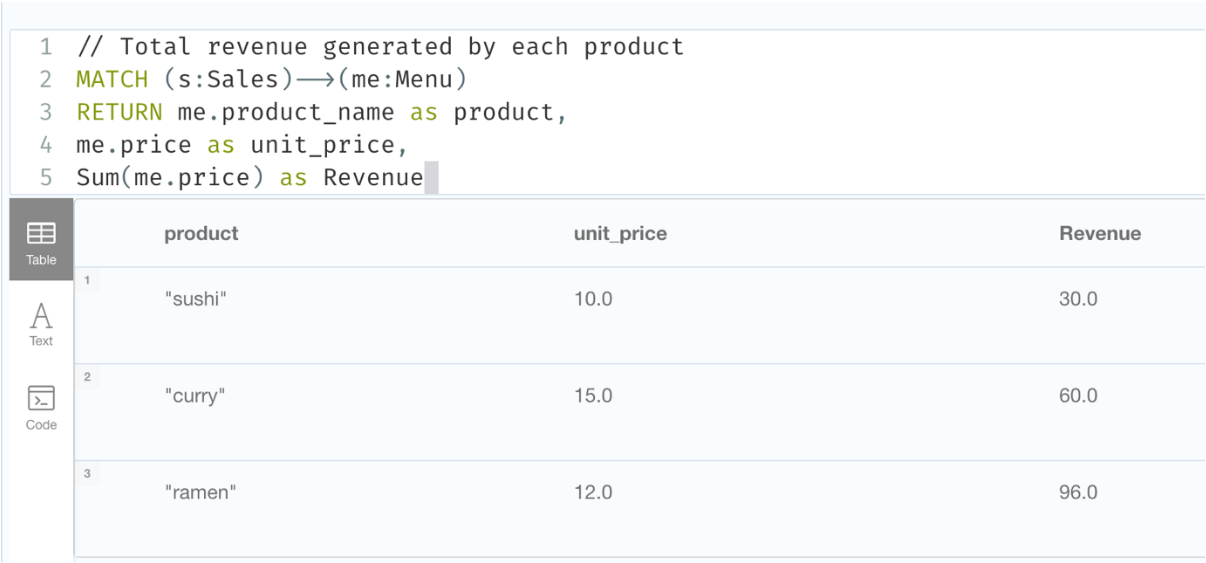
SQL→



```sql
SELECT
m.product_name AS Products,
m.price As 'Unit Price',
sum(price) As Revenue
From sales AS s
Join menu as m
on s.product_id= m.product_id
Group by m.product_name, m.price
```

Products	Unit Price	Revenue
sushi	10	30
curry	15	60
ramen	12	96

Cypher Query →

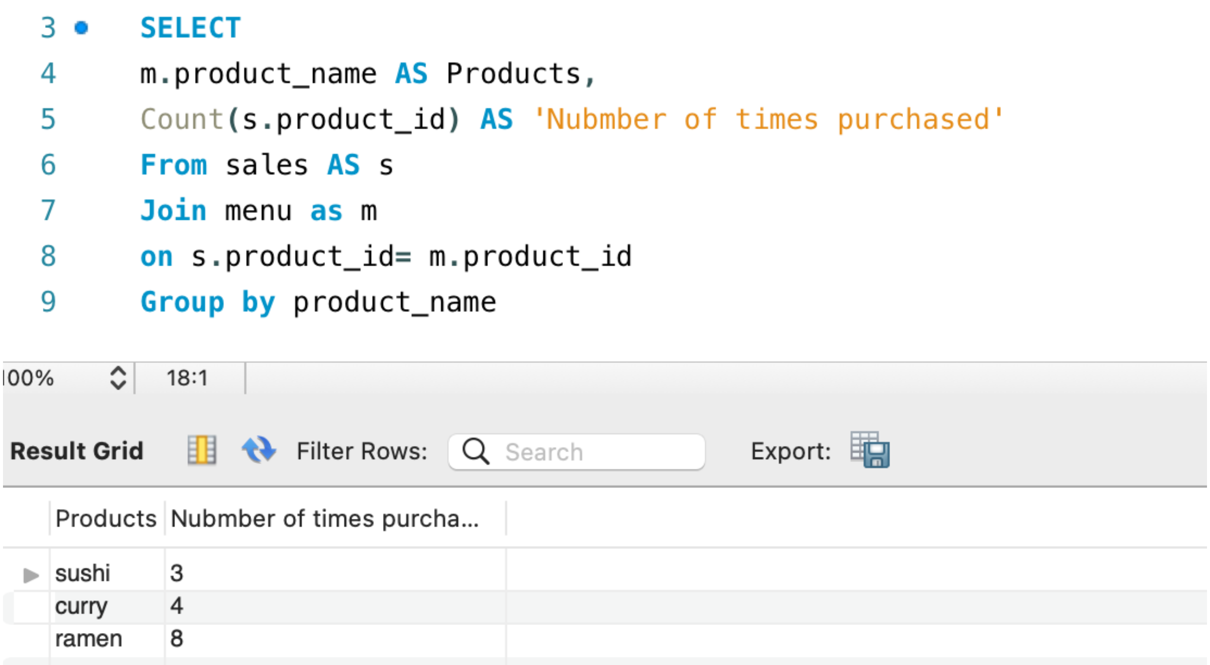


```
// Total revenue generated by each product
MATCH (s:Sales)-->(me:Menu)
RETURN me.product_name as product,
me.price as unit_price,
Sum(me.price) as Revenue
```

	product	unit_price	Revenue
1	"sushi"	10.0	30.0
2	"curry"	15.0	60.0
3	"ramen"	12.0	96.0

Question 4: Write a query to show the number of times each product was purchased.

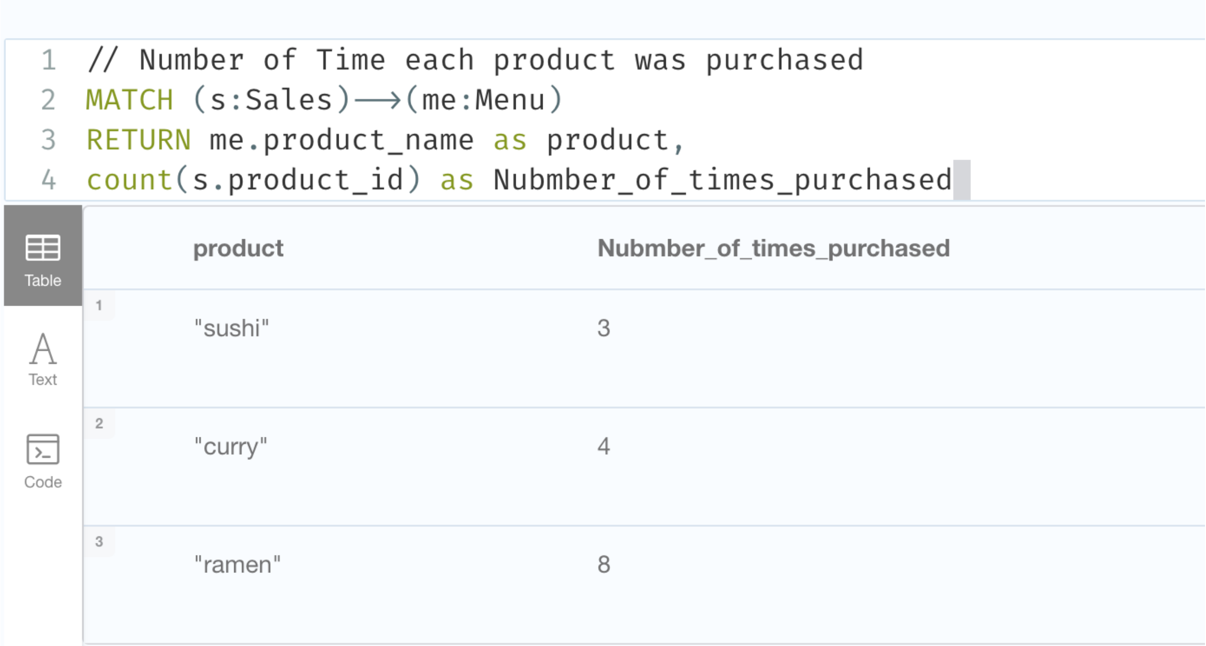
SQL→



```sql
SELECT
m.product_name AS Products,
Count(s.product_id) AS 'Nubmber of times purchased'
From sales AS s
Join menu as m
on s.product_id= m.product_id
Group by product_name
```

Products	Nubmber of times purcha...
sushi	3
curry	4
ramen	8

Cypher Query →

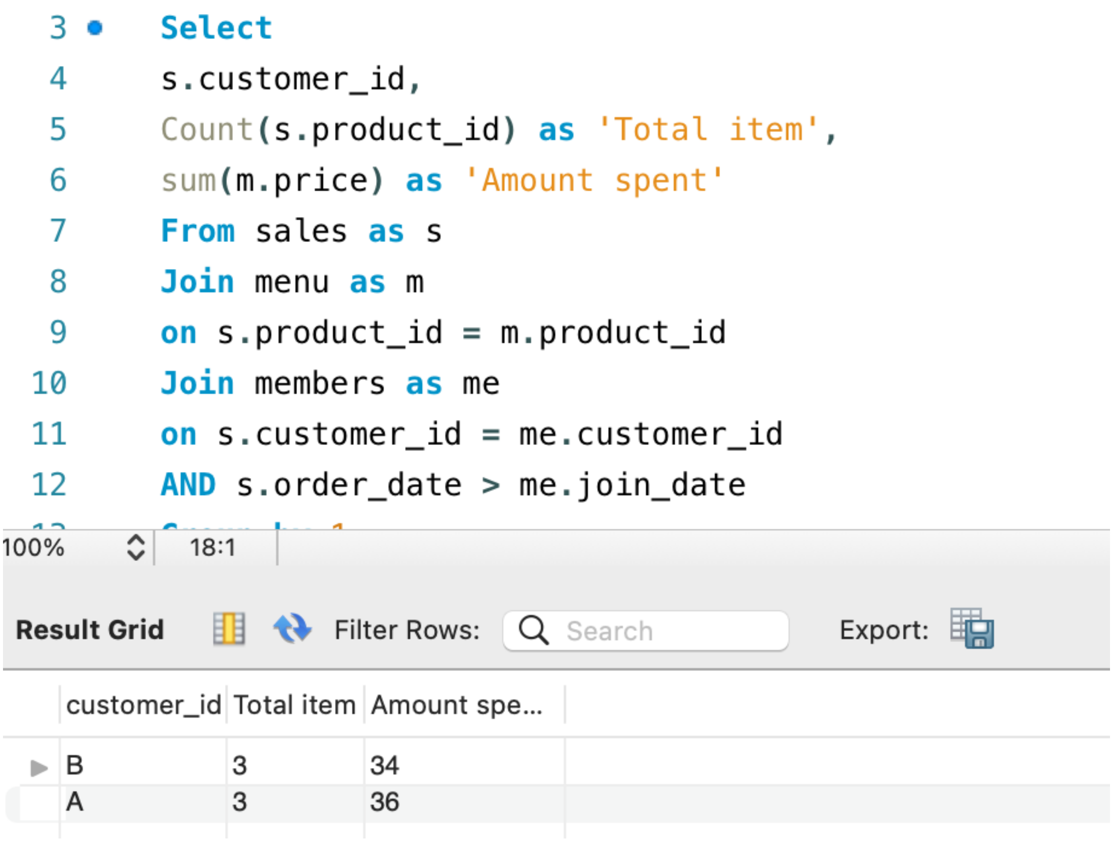


```
// Number of Time each product was purchased
MATCH (s:Sales)-->(me:Menu)
RETURN me.product_name as product,
count(s.product_id) as Nubmber_of_times_purchased
```

	product	Nubmber_of_times_purchased
1	"sushi"	3
2	"curry"	4
3	"ramen"	8

Question 5: Write a query to show the amount spent by each customer before they became a member.

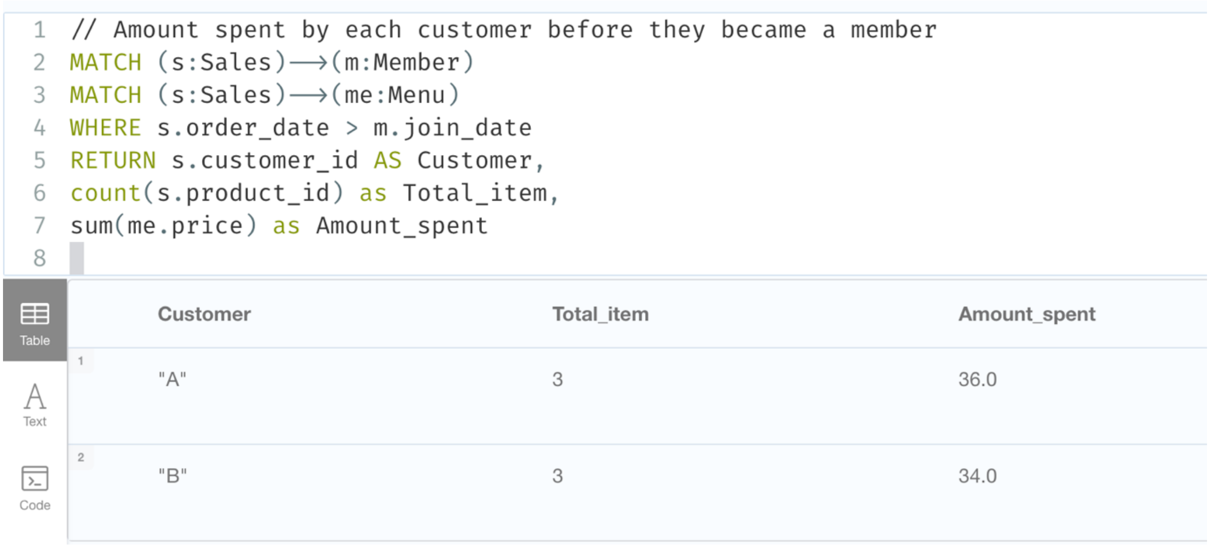
SQL→



```sql
Select
s.customer_id,
Count(s.product_id) as 'Total item',
sum(m.price) as 'Amount spent'
From sales as s
Join menu as m
on s.product_id = m.product_id
Join members as me
on s.customer_id = me.customer_id
AND s.order_date > me.join_date
```

customer_id	Total item	Amount spe...
B	3	34
A	3	36

Cypher Query→

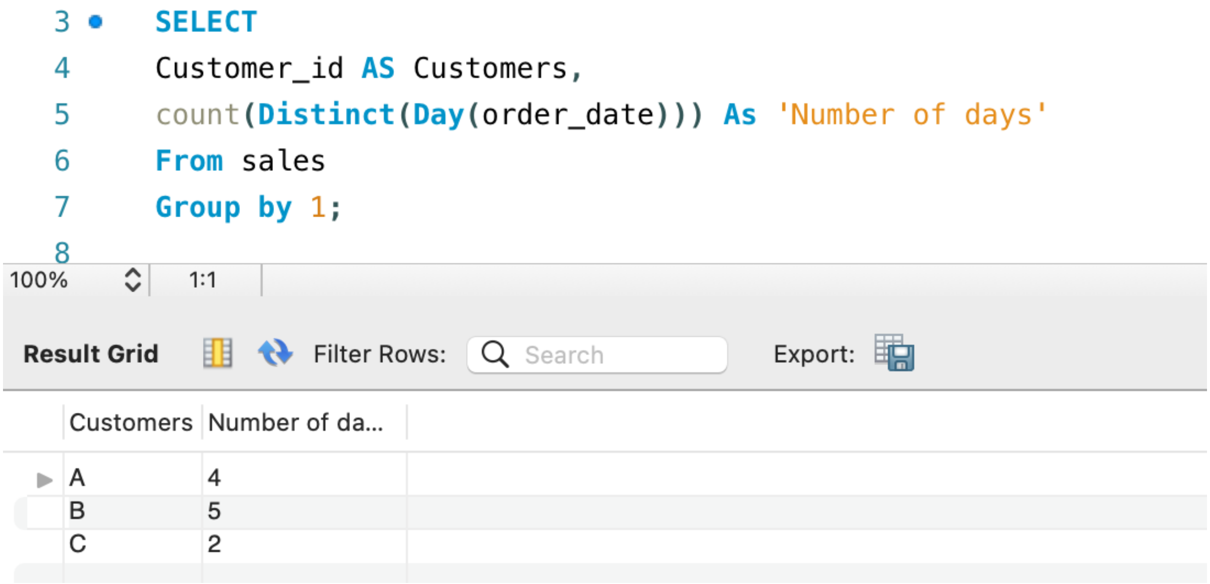


```
// Amount spent by each customer before they became a member
MATCH (s:Sales)-->(m:Member)
MATCH (s:Sales)-->(me:Menu)
WHERE s.order_date > m.join_date
RETURN s.customer_id AS Customer,
count(s.product_id) as Total_item,
sum(me.price) as Amount_spent
```

	Customer	Total_item	Amount_spent
1	"A"	3	36.0
2	"B"	3	34.0

Question 6: Write a query to show the number of days each customer visited the restaurant?

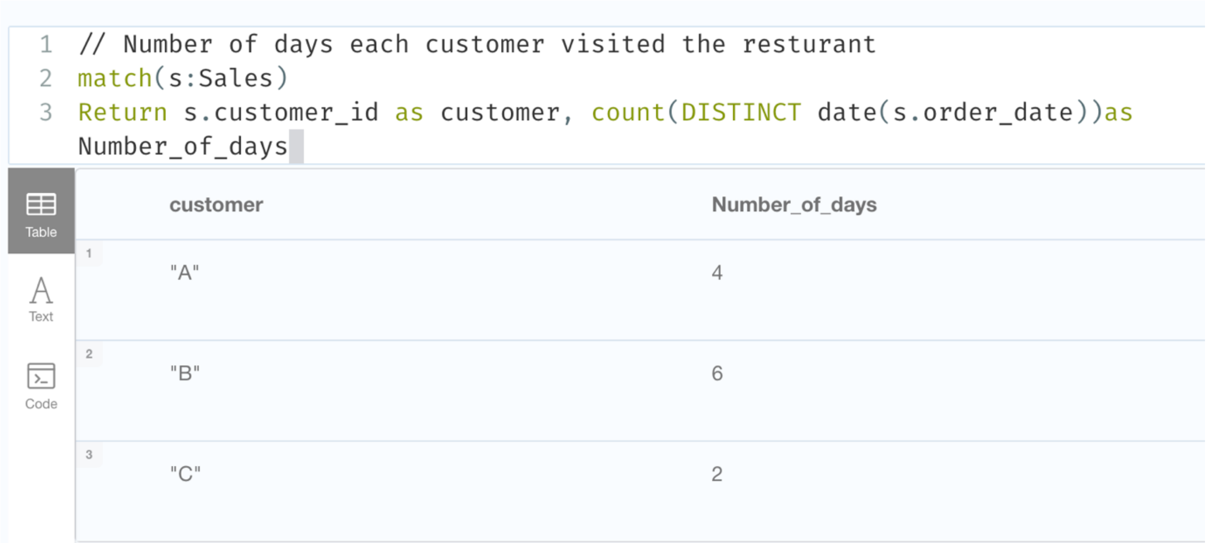
SQL →



```sql
SELECT
Customer_id AS Customers,
count(Distinct(Day(order_date))) As 'Number of days'
From sales
Group by 1;
```

Customers	Number of da...
A	4
B	5
C	2

Cypher Query →

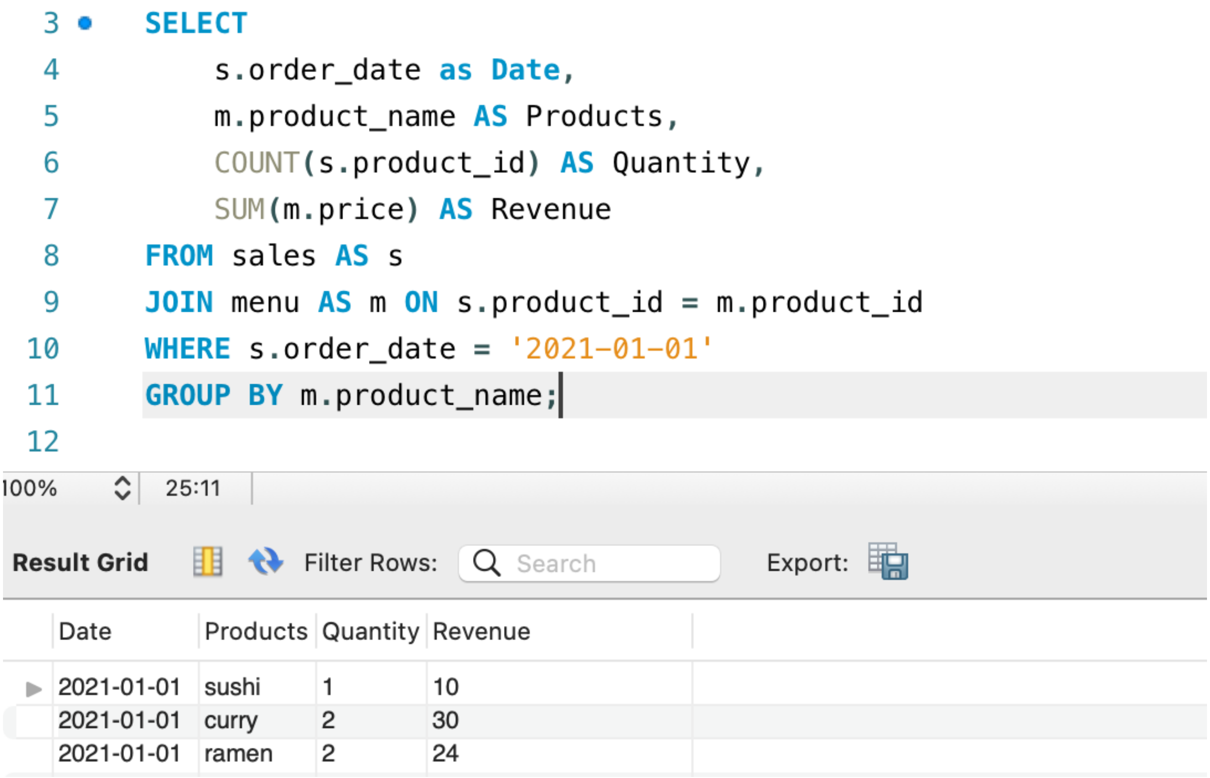


```
// Number of days each customer visited the resturant
match(s:Sales)
Return s.customer_id as customer, count(DISTINCT date(s.order_date))as
Number_of_days
```

	customer	Number_of_days
1	"A"	4
2	"B"	6
3	"C"	2

Question 7: Write a query to show the details of the products bought on the first day.

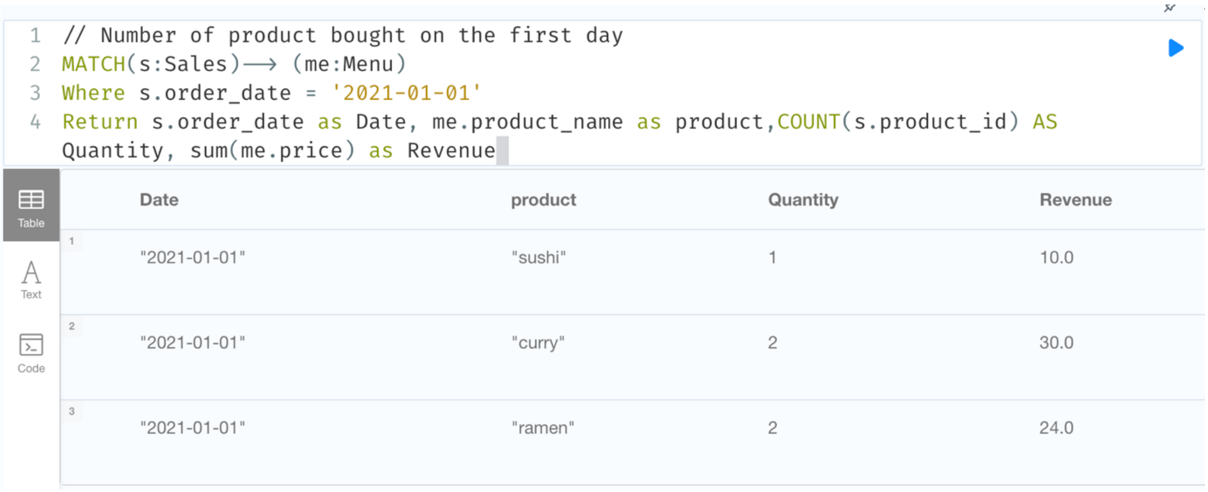
SQL →



```
SELECT
    s.order_date as Date,
    m.product_name AS Products,
    COUNT(s.product_id) AS Quantity,
    SUM(m.price) AS Revenue
FROM sales AS s
JOIN menu AS m ON s.product_id = m.product_id
WHERE s.order_date = '2021-01-01'
GROUP BY m.product_name;
```

Date	Products	Quantity	Revenue
2021-01-01	sushi	1	10
2021-01-01	curry	2	30
2021-01-01	ramen	2	24

Cypher Query →



```
// Number of product bought on the first day
MATCH(s:Sales)⟶ (me:Menu)
Where s.order_date = '2021-01-01'
Return s.order_date as Date, me.product_name as product,COUNT(s.product_id) AS
Quantity, sum(me.price) as Revenue
```

	Date	product	Quantity	Revenue
1	"2021-01-01"	"sushi"	1	10.0
2	"2021-01-01"	"curry"	2	30.0
3	"2021-01-01"	"ramen"	2	24.0

Conclusion

In this report we have successfully shown the process for creating a data warehouse, extracting, transforming and loading the data warehouse with relevant data to provide insights to stakeholders. We also went on to give an analytical report with visuals that are required for making informed decisions. Finally, we showed the difference between querying a structured relational database and querying a graph database.