MIDDLE EAST TECHNICAL UNIVERSITY

Retail Electronics and Home Appliances Management Database System

STAT 311 – Modern Database Systems

Final Project Report

Department of Statistics

Prepared by Group 2:

Gülizar Uz

Doğa Aksan

Rümeysa Daştan

Mert Cevdet Gürsel

Duhan Onat Karadayı

Instructor:

Deniz Çelikel

Course Assistant:

Mehmet Ali Erkan

(January 2026)

1. Abstract

This report explains how we designed and built a relational database system to manage retail operations for electronics and home appliances. The system handles key business tasks like managing customers, organizing products, processing orders, tracking payments, monitoring deliveries, and controlling inventory at different locations. We first created an Entity–Relationship (ER) model to show how the system works in the real world, then turned it into a logical relational schema.

Additionaly, we built the database using SQL and included primary and foreign key constraints to keep the data accurate and consistent. We added sample data to mimic real business situations and make analysis possible. With this schema, we can use SQL queries to answer research questions about sales, customer behavior in different regions, inventory levels, and how well the business runs.

We also created a web-based interface to make the system easy to use. Through this interface, users can see key performance indicators, view data visually, and run set queries on the database. Our project shows that a good database design, along with useful queries and a simple web interface, can help people make better decisions in retail.

2. Table of Contents

3. Introduction

Database systems are key to modern information systems because they let us store, find, and manage large amounts of data in an organized way. In real-world settings like retail, databases need to handle many types of information, such as customers, products, orders, payments, and inventory, while keeping everything consistent and reliable. A well-designed relational database is important for both daily work and analysis.

We developed a relational database system for a retail business specializing in electronics and home appliances. The system supports core functions, including customer registration, product catalog management, order processing, payment management, delivery tracking, inventory control, and service request handling. Our database design accurately models real-world relationships and enables efficient querying while maintaining data integrity.

As part of the Modern Database Systems course, this project focuses on using basic database concepts like entity–relationship modeling, designing relational schemas, and setting up primary and foreign key constraints to keep data linked correctly. We first made a conceptual design with an ER model, then turned it into a logical schema ready for a relational database system.

Besides designing the database, we also ran SQL queries to answer business questions and built a web-based interface for users to interact with the database. This combined approach shows that a well-organized database system can support both daily operations and analysis in retail.

4. Aim & Scope

The main goal of this project is to design and build a relational database system that accurately shows how a retail electronics and home appliances business works. We focus on turning real business needs into a clear and efficient database design that keeps data accurate, is easy to use, and supports analysis.

The project models essential business entities such as customers, products, categories, orders, order items, payments, deliveries, inventory, locations, stores, warehouses, and service requests. The relationships among these entities are defined using an entity-relationship (ER) model, and a corresponding relational schema is established with appropriate primary and foreign key constraints to maintain referential integrity and reliable data management.

We also prepared sample data to mimic real business situations. This data lets us run SQL queries to answer questions about sales, customer behavior, regional trends, inventory, and how well the business operates. Our analysis is limited to SQL queries and does not include advanced data mining or machine learning.

We also built a web-based interface so users can easily work with the database. This interface lets users run set queries, see key performance indicators, and view visual summaries, making the database more practical to use.

The project only covers designing and building a relational database as part of the Modern Database Systems course. We do not address performance tuning, security, or large-scale deployment in this study.

5. Database Design

In the database design stage, we turn business needs into an organized data structure. We start by creating a conceptual model and then move on to build a logical schema. The goal is to make sure the database reflects actual entities and their connections, keeps data accurate, and works. This part covers the conceptual design, the entity-relationship diagram, and the logical schema we used.

5.1 ER Model (Conceptual Design)

We created the conceptual design using an Entity–Relationship (ER) model to show the main parts of a retail electronics and home appliances business. The ER model lists the key entities in the system and explains how they are connected based on real business processes.

The Customer entity plays a key role in the model, standing for people who make purchases or ask for services. Customers connect to Orders, Payments, and Service Requests showing both their buying habits and service needs. An Order ties to one or more products via the Order Item entity. This entity keeps track of details like how many products are ordered and the price per unit.

The Product entity stands for items the business sells and is grouped under the Category entity to allow for a product hierarchy. Products are also connected to Inventory records, which show stock levels at different Locations. Locations are divided into Store and Warehouse entities, so the system can tell the difference between sales points and storage sites.

The ER model also includes operational processes. Payments and Deliveries are tied to Orders to record payments and shipping details. The Service Request entity handles customer support for specific products.

The ER model gives a high-level view of the system by clearly showing the entities, their attributes, and how they are related. This structure is the starting point for the logical schema and makes sure the database design fits real business needs.

5.2 ER Diagram (Visual Representation)

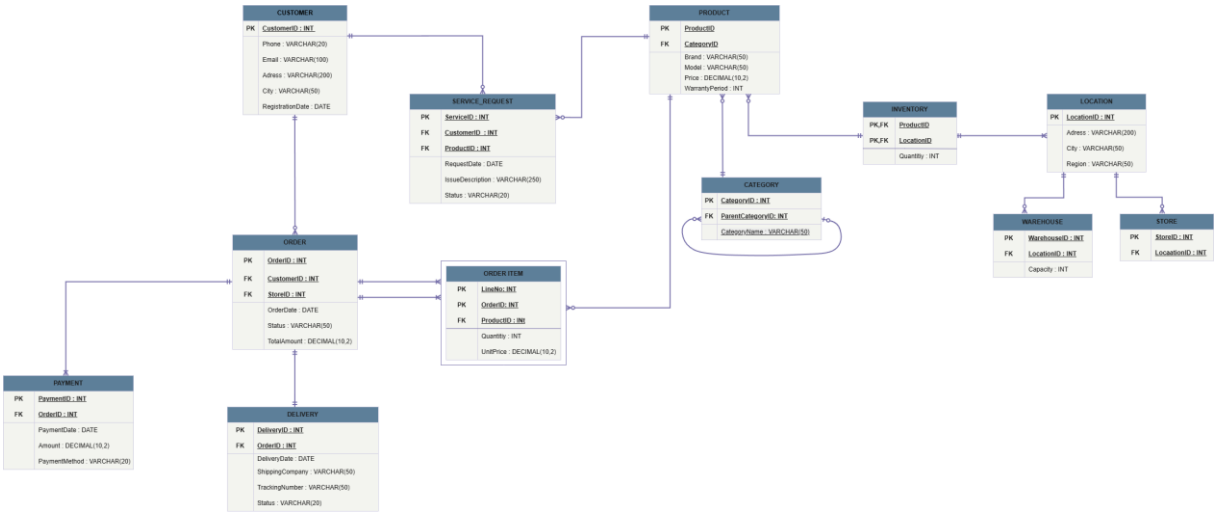


Figure 1. ER diagram of the retail electronics and home appliances database system.

It uses important structures like one-to-many and many-to-many relationships. Associative entities help explain complicated links between data. Primary and foreign keys play a role in creating data connections and keeping referential integrity intact. In general, the diagram acts as a clear visual representation of the database. It connects the main design idea to the detailed logical structure. The ER diagram illustrates the fundamental design of the system as previously described. It emphasizes the primary entities, their attributes, and the relationships among them. The diagram demonstrates the interactions between customers, products, orders, payments, deliveries, inventory, and locations within the retail system.

5.3 Logical Schema

In this section, we show the relational (logical) schema based on the ER model. Each entity becomes a table, and relationships are set up using foreign keys. Primary keys make each record unique, and foreign keys keep links between tables correct. This schema supports main retail tasks like customer registration, managing products, handling orders, payments, deliveries, tracking inventory at different locations, and after-sales service requests.

Customer (CustomerID, CustomerName, Phone, Email, Address, City, RegistrationDate)
CustomerID is the primary key. This table stores customer profiles and contact information.

Category (CategoryID, ParentCategoryID, CategoryName)
CategoryID is the primary key. ParentCategoryID is a foreign key that references CategoryID in the same table, allowing categories to be organized in a hierarchical structure where each category may have a parent. ParentCategoryID may be NULL for top-level categories.
Product (ProductID, CategoryID, Brand, Model, Price, WarrantyPeriod)

ProductID is the primary key. CategoryID is a foreign key to Category(CategoryID), linking each product to a category.

Location(LocationID, Address, City, Region)
LocationID is the primary key. This table stores physical location details shared by stores and warehouses.

Store (StoreID, LocationID)
StoreID is the primary key. LocationID is a foreign key to Location(LocationID), meaning each store is associated with one location.

Warehouse (WarehouseID, LocationID, Capacity)
WarehouseID is the primary key. LocationID is a foreign key to Location(LocationID), and Capacity stores the warehouse capacity attribute.

Orders (OrderID, CustomerID, StoreID, OrderDate, Status, TotalAmount)
OrderID is the primary key. CustomerID references Customer(CustomerID), and StoreID references Store(StoreID). This table represents a purchase event: a customer places an order at a specific store, with summary-level order attributes stored at the order header level.

Order_Item (LineNo, OrderID, ProductID, Quantity, UnitPrice)
Primary key is (LineNo, OrderID). OrderID is a foreign key to Orders(OrderID), and ProductID is a foreign key to Product(ProductID). This relation resolves the many-to-many relationship between Orders and Product by storing line-level order details. Each order can include multiple products, and each product can appear in many orders. LineNo uniquely identifies each item line within the same order.

Inventory (ProductID, LocationID, Quantity)
Primary key is (ProductID, LocationID). ProductID references Product(ProductID), and LocationID references Location(LocationID). This relation models stock levels of each product at each location. It resolves the many-to-many relationship between Product and Location by storing the current Quantity for each product-location pair.

Payment (PaymentID, OrderID, PaymentDate, Amount, PaymentMethod)
PaymentID is the primary key. OrderID is a foreign key to Orders(OrderID). This table stores payment records for orders, including the payment method and the amount paid.

Delivery (DeliveryID, OrderID, DeliveryDate, ShippingCompany, TrackingNumber, Status)
DeliveryID is the primary key. OrderID is a foreign key to Orders(OrderID). This table stores shipment/delivery details linked to an order, including carrier and tracking information.

Service_Request (ServiceID, CustomerID, ProductID, RequestDate, IssueDescription, Status)
ServiceID is the primary key. CustomerID references Customer(CustomerID), and ProductID references Product(ProductID). This table supports after-sales processes by recording customer-created service requests for specific products, including the request date, issue description, and current status.

The schema keeps master data (like Customer, Category, Product, Location, Store, Warehouse), transactional data (such as Orders, Order_Item, Payment, Delivery, Service_Request), and stock data (Inventory) separate. We use foreign keys to make sure all references are valid. For example, an order must have a real customer and store, and an order item must have a real order and product. Associative tables like Order_Item and Inventory help manage many-to-many relationships and make queries more efficient

6. Data Preparation & Assumptions

We created the data for this project to fit the database schema and simulate real retail operations. Instead of using real customer or business data, we made a synthetic dataset that shows common situations in electronics and home appliances retail. This way, we can test the database structure, relationships, and queries in a controlled and consistent way.

We added all records after setting up the full database schema. To keep references correct, we followed a specific order: first, we entered master data like customers, categories, products, locations, stores, and warehouses. Then we added transactional and operational data, such as orders, order items, payments, deliveries, inventory, and service requests. This order makes sure foreign key rules are followed and no invalid references are made.

We made several assumptions when building the dataset. Each order comes from one customer at one store, but it can include many products through the Order_Item table. Payment and delivery records always match real orders. Inventory numbers show current stock at each location and can change with sales or restocking. For product categories, top-level categories have a NULL ParentCategoryID.

The synthetic dataset is not meant for real-world economic or behavioral analysis. Its main purpose is to test the logical schema, check relational constraints, and allow useful SQL analysis for our research questions. The assumptions we made keep the dataset simple but still reflect the main structure and behavior of a real retail database.

7. Research Questions & SQL Analysis

RQ1 — Category preferences by geography

How do product category preferences differ across regions and cities?

Purpose

This question aims to analyze the different regions and cities, the demand for different product categories, and the relationship between units and revenue to analyze the overall value and the contributions of each location.

Data Structure Consideration

In the database, each order and each order item has a direct one-to-many relationship, which is represented in the database with the Order-Order Item relationship as an order table and an order item table. As a result, the demand for each category will require line-level detail as opposed to order-level summations, as well as the possibility of hierarchical parent-child category relationships, which require self-joins to retrieve parent categories from the Category table.

Involved Tables

- Orders
- Store
- Location
- Order_Item
- Product
- Category (and parent Category as pc)

Analytical Approach

For each order, the analysis is conducted by store and location. Each order line is linked to a product and its category, including parent categories. Results are then consolidated by region, city, and category. The analysis provides:

units_sold = total quantity of each category sold per location.

revenue = total of each category per location in value = quantity × unitPrice

Lastly, the results are sorted by the category each city received, highlighting each city's most valuable category.

SQL Query

```
SELECT
  l.region,
  l.city,
  COALESCE(pc.categoryname, 'NO_PARENT') AS parent_category,
  c.categoryname AS category,
  SUM(oi.quantity) AS units_sold,
  SUM(oi.quantity * oi.unitprice)::numeric(12,2) AS revenue
FROM orders o
JOIN store s        ON s.storeid = o.storeid
JOIN location l     ON l.locationid = s.locationid
JOIN order_item oi  ON oi.orderid = o.orderid
JOIN product p      ON p.productid = oi.productid
JOIN category c     ON c.categoryid = p.categoryid
LEFT JOIN category pc ON pc.categoryid = c.parentcategoryid
GROUP BY l.region, l.city, parent_category, c.categoryname
ORDER BY l.region, l.city, revenue DESC;
```

Query Output

*Table 1 presents the **category-level demand** by region and city, including parent category, units sold, and revenue.*

	region character varying (50)	city character varying (50)	parent_category character varying	category character varying (50)	units_sold bigint	revenue numeric (12,2)
1	Aegean Region	Aydın	Kitchen Appliances	Microwaves	1	14499.99
2	Aegean Region	Denizli	Small Appliances	Coffee Machines	3	22499.85
3	Aegean Region	İzmir	Small Appliances	Vacuum Cleaners	1	9999.49
4	Aegean Region	Manisa	Kitchen Appliances	Ovens	2	36999.80
5	Aegean Region	Muğla	Small Appliances	Blenders	2	12999.80
6	Black Sea Region	Ordu	Small Appliances	Electric Kettles	1	1899.90
7	Black Sea Region	Rize	Small Appliances	Toasters	3	7499.85
8	Black Sea Region	Samsun	Major Appliances	Water Dispensers	1	12499.90
9	Black Sea Region	Trabzon	Small Appliances	Hair Dryers	2	3999.98
10	Black Sea Region	Zonguldak	Major Appliances	Refrigerators	2	53999.98

Results & Interpretation

The data returned shows 30 unique combinations of region, city, and product category. For ease of understanding, the top 10 rows by revenue in the table are discussed in the primary report.

The report shows revenue data:

Major and Kitchen Appliances have the highest revenue in only a few cities, but very low unit sales. This suggests that the products are very expensive.

In the Small Appliances category, unit sales are much higher, but revenue is lower, indicating more purchases of lower-priced products.

Differences between regions are especially notable: the Aegean Region cities have a stronger demand for kitchen appliances, while the Black Sea Region cities have higher sales of small household appliances.

This data supports the conclusion that the range of product offerings and their relative price points vary by region, which is useful for developing targeted inventory and marketing plans.

RQ2 - Regional Differences in Average Order Value (AOV)

How does the average order value vary across regions and cities?

Purpose

This research question aims to understand the variation in average order value (AOV) across geographies. This helps reveal differences in basket size and purchasing frequency by region.

Data Structure Consideration

The databases' unique design records a single order as consisting of multiple order items via the Order_Item table. Thus, the total amount on an order reflects the total number of items and their unit prices. Hence, the differing AOV values reflect the varying basket sizes and purchase frequencies across cities.

Involved Tables

- Orders
- Store

- Location

Analytical Approach

Orders were aggregated separately for each region and city. The average total order amount was then calculated. To gauge confidence in the average, the order count (order_count) was included alongside AOV.

SQL Query

```
SELECT
    l.region,
    l.city,
    COUNT(*) AS order_count,
    ROUND(AVG(o.totalamount), 2) AS avg_order_value
FROM orders o
JOIN store s       ON s.storeid    = o.storeid
JOIN location l    ON l.locationid = s.locationid
GROUP BY
    l.region,
    l.city
ORDER BY
    avg_order_value DESC;
```

Query Output

*Table 2 presents the **average order value** by region and city, together with the number of orders used in the calculation.*

	region character varying (50)	city character varying (50)	order_count bigint	avg_order_value numeric
1	Mediterranean Region	Adana	1	83999.98
2	Marmara Region	Tekirdağ	1	65999.98
3	Marmara Region	Edirne	1	65999.85
4	Southeastern Anatolia Region	Gaziantep	1	56998.50
5	Marmara Region	Bursa	1	55999.98
6	Central Anatolia Region	Eskişehir	1	55498.50
7	Black Sea Region	Zonguldak	1	53999.98
8	Marmara Region	İstanbul	1	39999.90
9	Aegean Region	Manisa	1	36999.80
10	Eastern Anatolia Region	Van	1	34999.90

Results & Interpretation

The query output shows the distinct pairs of region and city, along with the order count and average order value (AOV), in 30 rows. The primary report shows the average order values for the top 10 cities to make reading easier.

Results show that:

Not all cities with higher-order AOV values also have more order counts. It can be assumed that the order count has a lesser effect on the AOV, since product pricing and basket size also contribute.

Cities in different regions exhibited high AOV with only one order recorded. This indicates that the orders involve a high quantity of items and/or high-value items.

Every region has a different average order value. The cities of the Marmara and Mediterranean Regions show the highest AOV, while some cities in the Aegean and Eastern Anatolia Regions show lower AOV.

The results show that the average order value differs significantly by region, driven by variation in basket size and customer order behavior, not just the number of orders. Such differences can be used to determine pricing and promotional limits for a region, the product marketing strategy, and the product combinations offered.

RQ3 - Top-Performing Stores (Sales Concentration by Location)

Which store locations generate the highest total sales revenue?

Purpose

The purpose of this research question is to identify the top-performing store locations by ranking stores based on their total sales revenue.

This analysis helps benchmark branch performance and supports business decisions such as marketing prioritization, inventory allocation, and investment planning.

Data Structure Consideration

Every record in the database is linked with a shop thanks to the storeid field. Because total sales at the store level can be impacted by how many orders there are and the size of each order, total sales and the number of orders in each count are included in the analysis.

In order to keep the numbers accurate, computations do not include in the total any orders that have been cancelled or returned.

Involved Tables

- Orders
- Store
- Location

Analytical Approach

This analysis looks at sales by store and location to see where our revenue is most concentrated. For each store, we calculate the total sales revenue by adding up all the order amounts, and we also include the number of orders for a fuller picture. Finally, we rank the

results by total sales—from highest to lowest—to pinpoint the top five performing stores in every region.

SQL Query

```
SELECT
  o.storeid,
  l.city,
  l.region,
  SUM(o.totalamount)::numeric(12,2) AS total_sales,
  COUNT(*) AS order_count
FROM orders o
JOIN store s    ON s.storeid = o.storeid
JOIN location l ON l.locationid = s.locationid
WHERE o.status NOT IN ('Cancelled','Returned')
GROUP BY o.storeid, l.city, l.region
ORDER BY total_sales DESC
LIMIT 5;
```

Query Output

*Table 3 presents the **top 5 store locations** ranked by their total sales revenue.*
For each store, the table includes its city, region, total sales amount, and order count, enabling comparison of both transaction volume and total revenue.

	storeid integer	city character varying (50)	region character varying (50)	total_sales numeric (12,2)	order_count bigint
1	12	Adana	Mediterranean Region	83999.98	1
2	2	Bursa	Marmara Region	55999.98	1
3	19	Eskişehir	Central Anatolia Region	55498.50	1
4	25	Zonguldak	Black Sea Region	53999.98	1
5	1	İstanbul	Marmara Region	39999.90	1

Results & Interpretation

The query output consists of 5 rows, each representing a unique store location with its corresponding city, region, total sales revenue, and order count.
For readability, only the top 5 stores ranked by total sales are displayed in the report.

As a result the report shows:

- Adana (med region), Bursa (Marmara), Eskişehir (Central Anatolia), Zonguldak (Black sea), and İstanbul (Marmara) have the top revenue sales.
- As each store has 1 order total sales must have resulted from 1 high-value order, not from sales volume.
- Thus the sales concentration must reflect that single order value was high.

- Also the high-value orders being purchased is spread across multiple regions.

In short these results have the top store sales revenue, but not all of them have volume orders.

This report shows a need to interpret top stores sales with other measures of sales with store rank and for store order sales revenue with total sales value to average order sales value

(AOV) of the stores, and to look for other stores that have sold more than three orders (AOV) for sales revenue.

RQ4 - Store-Level Cancellation and Return Patterns

Which stores exhibit higher cancellation and return volumes, and how do these volumes compare across stores?

Purpose

This research question evaluates store performance by identifying which stores have the highest volume of cancellations and returns.

Differences in cancellation and return rates among stores affect operational reliability, fulfillment quality, and customer outcomes.

Data Structure Consideration

The analysis uses order records with status attributes of Completed, Cancelled, or Returned.

Since stores can have many orders, cancellation and return rates should be measured as ratios rather than raw counts.

Stores with very few orders, such as one or two, may show extreme rates of 0% or 100%, which reflect isolated events rather than overall performance.

Involved Tables

- Orders
- Store
- Location

Analytical Approach

The system associates each order with its corresponding store and location details. It calculates the total number of cancelled and returned orders for each store. It then determines cancellation and return rates by comparing these figures to the total number of orders.

Finally, it ranks stores by cancellation and return rates to identify those with the highest values.

SQL Query

```
SELECT
  o.storeid,
  l.city,
  l.region,
  COUNT(*) AS total_orders,
  SUM(CASE WHEN o.status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled_orders,
  SUM(CASE WHEN o.status = 'Returned'  THEN 1 ELSE 0 END) AS returned_orders,
  (SUM(CASE WHEN o.status = 'Cancelled' THEN 1 ELSE 0 END)::numeric /
COUNT(*))::numeric(10,3) AS cancel_rate,
  (SUM(CASE WHEN o.status = 'Returned'  THEN 1 ELSE 0 END)::numeric /
COUNT(*))::numeric(10,3) AS return_rate
FROM orders o
JOIN store s    ON s.storeid = o.storeid
JOIN location l ON l.locationid = s.locationid
GROUP BY o.storeid, l.city, l.region
ORDER BY cancel_rate DESC, return_rate DESC
LIMIT 10;
```

Query Output

*Table 4 displays the **top 10 store locations** ranked by their cancellation and return rates.* Each row includes store ID, city, region, total orders, counts of cancelled and returned orders, and the corresponding rates.

	storeid integer	city character varying (50)	region character varying (50)	total_orders bigint	cancelled_orders bigint	returned_orders bigint	cancel_rate numeric (10,3)	return_rate numeric (10,3)
1	10	Muğla	Aegean Region	1	1	0	1.000	0.000
2	22	Trabzon	Black Sea Region	1	1	0	1.000	0.000
3	16	Ankara	Central Anatolia Region	1	1	0	1.000	0.000
4	4	Edirne	Marmara Region	1	1	0	1.000	0.000
5	28	Malatya	Eastern Anatolia Region	1	1	0	1.000	0.000
6	11	Antalya	Mediterranean Region	1	0	1	0.000	1.000
7	23	Ordu	Black Sea Region	1	0	1	0.000	1.000
8	29	Gaziantep	Southeastern Anatolia Region	1	0	1	0.000	1.000
9	5	Tekirdağ	Marmara Region	1	0	1	0.000	1.000
10	17	Konya	Central Anatolia Region	1	0	1	0.000	1.000

Results & Interpretation

The query result shows 10 rows, each showing a single store and its order outcome.

Here's what we can see:
Every store in the table had just one order (total_orders = 1) leading to cancellation or return rates of either 1.000 or 0.000. These extreme numbers reflect what happened with one order, not how the stores operate.

We see both cancellations and returns in several areas — including Aegean, Black Sea, Central Anatolia, Marmara, Eastern Anatolia, Mediterranean, and Southeastern Anatolia. This tells us that one-off problems happen in different parts of the country, not just in one specific area.

Because we're looking at such a small number of orders, this table shows us individual events rather than how stores perform over time.
All in all, these results show that one-time events have a big impact on the cancellation and return rates at the store level in this dataset.

To obtain more trustworthy insights, future studies should use a minimum order limit (like stores with 3 or more orders) and, if needed, sort by region or city. This will help determine whether the issue is specific to certain stores or widespread across an area.

RQ5 - Brand/Model Sales Concentration

Which brands and models generate the highest sales volume and revenue?

Purpose

The aim of this research question is to identify the brands and product models that contribute most to total sales.

By ranking products by both **units sold** and **revenue**, this analysis highlights which models drive overall sales performance and whether high revenue is driven by product pricing or sales volume.

Data Structure Consideration

Each order may include multiple products, and each product belongs to a specific **brand** and **model** in the Product table.

Because revenue depends on both **quantity sold (oi.quantity)** and **unit price (oi.unitprice)**, total sales performance is calculated at the brand–model level.
In cases with very few orders, high revenue may result from a **single high-value purchase** rather than consistent demand.

Involved Tables

- Order_Item
- Product

Analytical Approach

The analysis aggregates sales data by **brand and model.**For each product model, the total orders, units sold, and revenue are calculated.

Results are ranked by total revenue, with units sold as a secondary criterion, and the **top 10 models** are presented.

SQL Query

```
SELECT
  p.brand,
  p.model,
  COUNT(DISTINCT oi.orderid) AS orders_count,
  SUM(oi.quantity) AS units_sold,
  SUM(oi.quantity * oi.unitprice)::numeric(12,2) AS revenue
FROM order_item oi
JOIN product p ON p.productid = oi.productid
GROUP BY p.brand, p.model
ORDER BY revenue DESC, units_sold DESC
LIMIT 10;
```

Query Output

*Table 5 presents the **top 10 brand–model combinations** ranked by total revenue, along with the corresponding number of orders and units sold.*
This enables the identification of which product lines dominate sales performance within the dataset.

	brand character varying (50)	model character varying (50)	orders_count bigint	units_sold bigint	revenue numeric (12,2)
1	Apple	iPhone 14	1	2	83999.98
2	Samsung	QLED Vision 55	1	2	65999.98
3	Vestel	CoolPro 18000BTU	1	3	65999.85
4	Philips	Ambilight 50	1	3	56998.50
5	Arcelik	AquaWash 9kg	1	2	55999.98
6	Arcelik	DeepFreeze 300	1	3	55498.50
7	LG	InstaFresh 450	1	2	53999.98
8	Beko	FrostFree X500	1	1	39999.90
9	Siemens	BakeMaster 700	1	2	36999.80
10	Vestel	CleanPro 12	1	2	34999.90

Results & Interpretation

The query returns 10 rows of data, each presenting a brand and product model along with order count, units sold, and total revenue.

Key observations include the following:

- Models with high earnings, such as the Apple iPhone 14 and Samsung QLED Vision 55, rank at the top. Although each has only one order, their high prices result in leading revenue figures rather than high sales volume.

- Several models from Turkish brands like Vestel, Arçelik, Beko, and Siemens are on the list. This shows that Turkish electronics and appliance makers play a significant role in large-value sales.
- Various product types, such as smartphones, TVs, air conditioners, washing machines, refrigerators, and ovens, indicate that expensive sales occur across several categories rather than being focused on just one.
- All mentioned models are associated with a single order, indicating that the results reflect revenue spikes from individual transactions rather than consistent sales patterns.

In summary, the analysis shows that most sales revenue comes from expensive items and single high-value orders. To better understand overall demand, future research should look at trends over time and how unit sales are spread across product categories.

RQ6 - After-Sales Service Concentration by Product Category

Which product categories generate the highest after-sales service workload?

Purpose

This case study attempts to identify product categories that have the highest number of requests for after-sales services. Once established, the categories will aid in improving the overall service quality, service warranty management, and service demand adjustment at a strategic level.

Data Structure Consideration

Each service request in the Service_Request table is linked to a specific product in the Product table. Each product is further classified within a broader group in the Category table. This structure enables aggregating service request data by category rather than by individual product, providing a more comprehensive view of maintenance and reliability trends.

Involved Tables

- Service_Request
- Product
- Category

Analytical Approach

The analysis calculates how many service requests are linked to each product category. It sorts the results by the number of service requests from highest to lowest and shows the five categories with the most demand for service. This method shows which products create the biggest strain on the after-sales service network.

SQL Query

```
SELECT
  c.categoryname,
  COUNT(sr.serviceid) AS service_request_count
```

```
FROM service_request sr
JOIN product p  ON p.productid = sr.productid
JOIN category c ON c.categoryid = p.categoryid
GROUP BY c.categoryname
ORDER BY service_request_count DESC
LIMIT 5;
```

Query Output

*Table 6 presents the **top 5 product categories** ranked by the total number of after-sales service requests received.*
Each row shows the category name and its corresponding count of service requests.

	categoryname character varying (50)	service_request_count bigint
1	Washing Machines	4
2	Air Conditioners	4
3	Vacuum Cleaners	4
4	Cooktops	2
5	Dryers	2

Results & Interpretation

The query output shows five rows, with each row standing for a product category and its total service request count.

Looking at the results:

- Washing Machines, Air Conditioners, and Vacuum Cleaners all have the most service requests at 4 each, hinting at more frequent after-sales problems in these categories.
- Cooktops and Dryers come next, each with 2 requests showing a fair amount of service needs.
- The data reveals that big household appliances make up most of the service workload. This matches the fact that they are more complex and used more often.

In general, these findings show that the main types of household appliances create a larger share of the after-sales service workload. This understanding helps the company plan maintenance resources, manage warranty expenses, and improve quality checks. It also supports better decisions about how to distribute service efforts across different appliance types.

RQ7 - Payment Method and Order Value Relationship

Does the average order value (AOV) differ by payment method?

Purpose

This research question aims to find out if customers spend more or less depending on their payment method. Knowing the average order value (AOV) for each payment type helps set pricing and promotions, design better checkout experiences, and decide on payment channel fees or incentives.

Data Structure Consideration

Each payment record connects to an order using the order ID. If one order involves several payment rows, like split or partial payments, it is important to calculate averages per order to avoid counting the same order multiple times. Cancelled or returned orders should not be included since they do not represent actual sales.

Involved Tables

- Payment
- Orders

Analytical Approach

Orders are matched with their payments and grouped by payment methods. For every method, we show how many orders were paid and calculate the AOV. The results are sorted by AOV to highlight payment methods tied to higher-value transactions.

SQL Query

```
SELECT
  p.paymentmethod,
  COUNT(DISTINCT o.orderid) AS payment_count,
  AVG(o.totalamount)::numeric(12,2) AS avg_order_value
FROM payment p
JOIN orders o ON o.orderid = p.orderid
WHERE o.status NOT IN ('Cancelled','Returned')
GROUP BY p.paymentmethod
ORDER BY avg_order_value DESC;
```

Query Output

Table 7 presents AOV by payment method together with the number of paid orders per method.

	paymentmethod character varying (20)	payment_count bigint	avg_order_value numeric
1	Cash	6	40499.40
2	Debit Card	6	37833.26
3	Installment	6	32333.26
4	Credit Card	6	21833.18
5	Bank Transfer	6	11983.15

Results & Interpretation

The table shows five different ways to pay each with six orders (payment_count = 6). This balance makes it easy to compare their average order values (AOV).

- Cash has the highest AOV at about 40,499.40. Debit Card comes next at 37,833.26, and Installment follows at 32,333.26.
- Credit Card demonstrates an average AOV of approximately 21,833.18, whereas Bank Transfer records the lowest AOV at 11,983.15.
- Because each payment method is associated with an equal number of orders, differences in AOV reflect variations in basket size or price, rather than frequency of use.
- Cash and debit cards show a higher average order value, which indicates people often use them to make pricey purchases. On the other hand, Bank Transfer connects to cheaper orders in this data.

In summary, the data reveals that people spend more using Cash and Debit Cards, while Installment and Bank Transfer transactions are tied to lower-value purchases. To make the most of this, businesses can focus on unique promotions or checkout features to push higher-value payments. For example, they could offer free shipping or upselling bundles to those who use Cash or Debit Cards. They should also find the right balance between installment fees and the value it gives to customers. checking trends in average order values across various areas and customer groups helps in updating prices and promotional plans.

RQ8 - Delivery Speed Differences Across Shipping Companies

Do shipping companies differ in their average delivery time?

Purpose

This research question looks at how shipping companies perform by checking their average delivery times. By comparing these times, we can find the fastest carriers, choose reliable partners, improve customer satisfaction, and plan better logistics.

Data Structure Consideration

Each order is associated with a delivery record in the Delivery table, which contains the shipping company, order placement date, and delivery date. Delivery time is calculated by subtracting the order date from the delivery date. To ensure data accuracy, cancelled or returned orders are excluded, and it is confirmed that the delivery date occurs after the order date

Involved Tables

- Delivery
- Orders

Analytical Approach

The system combines orders with delivery details to figure out the average delivery time in days for every shipping company. They also calculate how many deliveries each company completed, called the delivery count, to make the comparisons fair.

In the end, they organize the data by average delivery time. This helps show differences in how fast and reliable each carrier is.

SQL Query

```
SELECT
  d.shippingcompany,
  COUNT(*) AS delivery_count,
  ROUND(AVG(d.deliverydate - o.orderdate), 2) AS avg_days
FROM delivery d
JOIN orders o ON o.orderid = d.orderid
WHERE o.status NOT IN ('Cancelled', 'Returned')
  AND d.deliverydate IS NOT NULL
  AND d.deliverydate >= o.orderdate
GROUP BY d.shippingcompany
ORDER BY avg_days DESC;
```

Query Output

*Table 8 presents the **average delivery time** (in days) for each shipping company, along with the total number of completed deliveries per company.*
The table allows for comparison of delivery speed and consistency across logistics providers.

	shippingcompany character varying (50)	delivery_count bigint	avg_days numeric
1	Surat Cargo	6	3.00
2	Yurtici Cargo	6	3.00
3	PTT Cargo	6	3.00
4	MNG Cargo	6	3.00
5	Aras Cargo	6	3.00

Results & Interpretation

The final report includes five rows. Each row represents one shipping company, along with its delivery count and average delivery time.

The results show that all five companies — Surat Cargo, Yurtiçi Cargo, PTT Cargo, MNG Cargo, and Aras Cargo — had an average delivery time of 3 days, and each company completed six deliveries.

- The results show no clear difference in average delivery performance between the carriers included in this dataset.

- This consistency among the companies comes from a combination of factors. It could be due to a small and balanced sample size similar delivery processes, or even simulated data instead of actual real-world operations.

Overall, the data suggests that all the shipping companies in the sample achieve the same delivery speeds. To better understand these patterns future studies should analyze bigger datasets, look at variations by region or time, and examine performance differences to uncover subtle trends in carrier efficiency and dependability.

RQ9 - Regional Inventory Concentration

How is total inventory distributed across regions, and which regions hold the largest stock levels?

Purpose

This study examines the way inventory is distributed over various locations to figure out where the highest stock levels are. The results assist in planning warehouses, enhancing delivery systems, and maintaining a steady inventory. Having too much stock in certain areas might point to overstocking troubles or problems in the supply chain.

Data Structure Consideration

Each entry in the Inventory table shows how much stock is kept at a specific place, which connects to the Location table. Since a region can have several locations, the total stock for that region comes from adding up the quantities from all its locations. This setup helps see how inventory is spread out on a bigger (regional) level instead of focusing on single warehouses.

Involved Tables

- Inventory
- Location

Analytical Approach

To figure this out, they add up the stock amounts (quantity) in each region and count how many unique warehouses or stores (location_count) there are. This gives a clearer picture of how the stock is spread out. They arrange the results by total stock to highlight which regions have the most inventory.

SQL Query

```
SELECT
  l.region,
  COUNT(DISTINCT i.locationid) AS location_count,
  SUM(i.quantity) AS total_stock
FROM inventory i
JOIN location l ON l.locationid = i.locationid
GROUP BY l.region
```

```
ORDER BY total_stock DESC;
```

Query Output

*Table 9 presents **total stock levels** aggregated by region, along with the number of active inventory locations in each region.*
This allows identification of which regions store the highest proportions of total inventory.

	region character varying (50)	location_count bigint	total_stock bigint
1	Mediterranean Region	5	605
2	Black Sea Region	5	555
3	Aegean Region	5	530
4	Eastern Anatolia Region	3	417
5	Central Anatolia Region	5	380
6	Marmara Region	5	355
7	Southeastern Anatolia Region	2	193

Results & Interpretation

The query result shows 7 rows. Each one lists a region, its total stock, and the number of locations holding that inventory.

The information reveals that:

- The Mediterranean Region has the most stock at 605 units followed by the Black Sea with 555 units and the Aegean Region with 530 units.
- Southeastern Anatolia Region sits at the bottom with just 193 units of stock and two storage points, which hints at limited storage or sales options there.
- Eastern Anatolia, with 417 units, and Central Anatolia, with 380 units, show moderate stock levels. This points to a balanced but smaller inventory scale in these areas.
- Most regions have about five locations each, yet the stock amounts vary. This reveals that the inventory is spread, with some locations holding more items than others.

These observations indicate that the Mediterranean, Black Sea, and Aegean regions have the bulk of the inventory. This shows either stronger sales in these areas or a centralized approach to logistics. To improve the supply chain and cut storage costs future plans should focus on moving excess stock to regions that have less invento

RQ10 - Product-Based Low-Stock Locations (Threshold Analysis)

*Which products fall below the low-stock threshold (e.g., fewer than **85 units**), and in which locations do these shortages occur?*

Purpose

This research question aims to identify the risk of stock-outs by highlighting product-location pairs where inventory is below a set threshold. The results help plan restocking, facilitate transfers between stores, and decide safety stock or reorder levels.

Data Structure Consideration

The Inventory table tracks stock at the product and location level. It connects to product details like brand and model and geographic details like city and region through the Product and Location tables. Since early warnings are the goal, the query shows the current stock quantity (i.quantity) and checks it against a set threshold, which in this case is 85 units.

Involved Tables

- Inventory
- Product
- Location

Analytical Approach

Focus on inventory rows where the quantity is below the threshold. Add product and location details to the data, then arrange it in ascending order by stock levels to highlight the biggest shortages first. You can also assign severity levels like Critical for less than 60 units, High for 60 to 74 units, and Moderate for 75 to 84 units.

SQL Query

```
SELECT
    p.productid,
    p.brand,
    p.model,
    l.locationid,
    l.city,
    l.region,
    i.quantity AS stock_qty
FROM inventory i
JOIN product  p ON p.productid  = i.productid
JOIN location l ON l.locationid = i.locationid
WHERE i.quantity < 85

ORDER BY i.quantity ASC, p.productid;
```

Query Output

*Table 10 lists product–location pairs under the **85-unit** threshold, including product ID, brand, model, location (ID, city, region), and current stock quantity.*

	productid integer	brand character varying (50)	model character varying (50)	locationid integer	city character varying (50)	region character varying (50)	stock_qty integer
1	29	Philips	Ambilight 50	27	Gaziantep	Southeastern Anatolia Region	53
2	15	Braun	SteamIron Pro	32	Isparta	Mediterranean Region	55
3	1	Beko	FrostFree X500	34	İstanbul	Marmara Region	57
4	16	Rowenta	EcoHeater 2000	6	Ankara	Central Anatolia Region	62
5	2	Arcelik	AquaWash 9kg	16	Bursa	Marmara Region	64
6	17	JBL	CinemaSound 5.1	42	Konya	Central Anatolia Region	69
7	3	Bosch	SilentClean 60	41	Kocaeli	Marmara Region	71
8	18	Beko	DryPlus 8kg	38	Kayseri	Central Anatolia Region	76
9	4	Vestel	CoolPro 18000BTU	22	Edirne	Marmara Region	78
10	19	Arcelik	DeepFreeze 300	26	Eskişehir	Central Anatolia Region	83

Results & Interpretation

The query brings back 10 rows. Each row shows a specific product with low stock in one location. Stock numbers fall between 53 and 83 units across regions like Southeastern Anatolia, Mediterranean, Marmara, and Central Anatolia.

The findings reveal that:

- Many important home appliances and electronics—like Philips Ambilight 50, Braun SteamIron Pro, Beko FrostFree X500, Arçelik AquaWash 9kg / DeepFreeze 300, and Vestel CoolPro 18000BTU—are running out of stock in key cities such as Gaziantep, İstanbul, Bursa, Edirne, and Eskişehir.
- The limited stock is spread out across different areas, so it is better to focus on restocking specific cities instead of trying to cover entire regions. Products that have 60 or fewer units in stock (like 53 to 60) show a greater risk right now and need restocking as soon as possible.

In general, the results indicate businesses need to focus on proactive restocking to avoid running out of supplies. Companies should set up systems to restock inventory or transfer items between locations when necessary. They should check safety stock limits and revise reorder point calculations, such as using the formula ($ROP = demand\_during\_leadtime + z \cdot \sigma\_LT$). Monitoring how products sell across different locations can help adjust the 85-unit limit for each item and area as needed.

RQ11 - Proactive Customer Outreach for Problematic Products

Which customers purchased the product(s) with the highest number of service requests, and what are their contact details (phone/email) for targeted post-issue communication and compensation campaigns?

Purpose

This research question aims to find the customers who bought the most troublesome products. These are products linked to the greatest number of service requests. The goal is to help the company reach out to these customers to offer specific support, launch product recall efforts, or provide goodwill compensation.

Reaching out to these customers ahead of time can help the business boost customer satisfaction, build trust in the brand, and make service processes more effective.

Data Structure Consideration

The Service_Request table keeps service data tied to each product. Customer purchases connect to this data using Orders and Order_Item. By combining these sets of data, it becomes clear which customers are impacted by issues with product reliability. Even if a customer has bought several problematic products, they will show up once they are identified.

Involved Tables

- Customer
- Orders
- Order_Item
- Service_Request

Analytical Approach

This analytical approach identifies the products with the highest number of service requests and then finds the unique customers who purchased these items. It retrieves each customer's contact details, phone number, email, and city to support targeted communication such as follow-ups or service notifications. To maintain a professional and readable report layout, the results are limited to the first 10 customers, ensuring clarity without unnecessary complexity.

SQL Query

```
SELECT DISTINCT
  c.customerid,
  c.customername,
  c.phone,
  c.email,
  c.city
FROM customer c
JOIN orders o      ON o.customerid = c.customerid
JOIN order_item oi ON oi.orderid   = o.orderid
WHERE o.status NOT IN ('Cancelled','Returned')
  AND oi.productid IN (
    SELECT sr.productid
    FROM service_request sr
    GROUP BY sr.productid
    HAVING COUNT(*) = (
      SELECT MAX(cnt)
      FROM (
        SELECT COUNT(*) AS cnt
        FROM service_request
        GROUP BY productid
      ) AS t
```

```
    )
  )
ORDER BY c.customerid
LIMIT 10;
```

Query Output

*Table 11 lists the **first 10 customers** who purchased the product(s) with the highest service request count, including their **contact details** for potential follow-up.*

	customerid [PK] integer	customername character varying (50)	phone character varying (20)	email character varying (100)	city character varying (50)
1	2	Ayşe Kaya	+90 532 102 1222	ayse.kaya@example.com	Bursa
2	4	Elif Şahin	+90 534 104 1424	elif.sahin@example.com	Edirne
3	6	Zeynep Aydın	+90 536 106 1626	zeynep.aydin@example.com	İzmir
4	8	Fatma Koç	+90 538 108 1828	fatma.koc@example.com	Aydın
5	10	Emine Öztürk	+90 530 110 2030	emine.ozturk@example.com	Muğla
6	12	Hakan Kılıç	+90 532 112 2232	hakan.kilic@example.com	Adana
7	14	Burak Aksoy	+90 534 114 2434	burak.aksoy@example.com	Hatay
8	16	Serkan Doğan	+90 536 116 2636	serkan.dogan@example.com	Ankara
9	18	Onur Eren	+90 538 118 2838	onur.eren@example.com	Kayseri
10	20	Oğuzhan Yıldırım	+90 530 120 3040	oguzhan.yildirim@example.com	Nevşehir

Results & Interpretation

The query results show 10 customers who purchased at least one of the problematic products, based on how service requests were logged.

The findings highlight that:

- The customers affected come from many different areas, including cities like Bursa, Edirne, İzmir, Aydın, Adana, Hatay, Ankara, Kayseri, Nevşehir, and Muğla.
- You can reach out to each customer either by phone or email, which allows one-on-one communication and services designed to meet their individual needs.
- Because the customers are spread across different regions, it is a good idea to use a central team to work with local service teams to handle communication and follow-ups more.
- Taking proactive steps, like offering extended warranties, providing replacement options, or sending apology messages, can repair customer trust and ease concerns.

These findings show that reaching out to customers allows the company to change possible service problems into chances to improve customer care. This method helps lower the risk of losing customers and builds trust in the brand while boosting its reputation after sales.

RQ12 - Least-Selling Products by Top-Level Category

For each top-level product category, which product has the lowest sales volume (in units sold)?

Purpose

Find the product with the lowest sales in each main category to spot potential gaps in the range or to decide if the item should be marked down, bundled, or removed.

Data Structure Consideration

Each product falls under a category, which might be part of a parent–child structure. To group them at the highest level, use COALESCE(parent.categoryname, category.categoryname).

Measure the sales volume using the order quantities (Order_Item.quantity), ensuring to leave out any cancelled or returned orders.

Involved Tables

- Product
- Category (and self-join as parent)
- Order_Item
- Orders (to filter invalid statuses)

Analytical Approach

Add up the total units sold for every product in every top-level category. Sort the products by the number of units sold, starting with the least. Pick the product with the lowest sales (including any ties) in each category.

SQL Query

```
SELECT
  COALESCE(pc.categoryname, c.categoryname) AS top_category,
  p.productid,
  p.brand,
  p.model,
  SUM(oi.quantity) AS units_sold
FROM orders o
JOIN order_item oi ON oi.orderid   = o.orderid
JOIN product     p ON p.productid  = oi.productid
JOIN category    c ON c.categoryid = p.categoryid
LEFT JOIN category pc ON pc.categoryid = c.parentcategoryid
WHERE o.status NOT IN ('Cancelled','Returned')
GROUP BY COALESCE(pc.categoryname, c.categoryname),
         p.productid, p.brand, p.model
ORDER BY top_category, units_sold ASC
LIMIT 10;
```

Query Output

*Table 12 shows the **least-selling products** per top-level category (recommended query), or the **first 10 low-volume products** across categories (alternative), including top_category, productid, brand, model, and units_sold.*

	top_category character varying (50)	productid [PK] integer	brand character varying (50)	model character varying (50)	units_sold bigint
1	Climate Control	28	Beko	CoolSmart 12000BTU	1
2	Climate Control	16	Rowenta	EcoHeater 2000	1
3	Climate Control	14	Xiaomi	PureAir 4	3
4	Climate Control	4	Vestel	CoolPro 18000BTU	3
5	Entertainment	5	Samsung	QLED Vision 55	2
6	Entertainment	17	JBL	CinemaSound 5.1	2
7	Entertainment	29	Philips	Ambilight 50	3
8	Kitchen Appliances	8	LG	QuickWave 25L	1
9	Kitchen Appliances	3	Bosch	SilentClean 60	1
10	Kitchen Appliances	27	Vestel	CleanPro 12	2

Results & Interpretation

The query output points out items with low sales volumes from various top-level categories. These include Climate Control products like *Beko CoolSmart 12000BTU, Rowenta EcoHeater 2000, Xiaomi PureAir 4, Vestel CoolPro 18000BTU*. It also lists Entertainment devices such as *Samsung QLED Vision 55, JBL CinemaSound 5.1, Philips Ambilight 50*. The list features Kitchen Appliances like *LG QuickWave 25L, Bosch SilentClean 60, Vestel CleanPro 12*.

The findings show that:

- Many products on the list have sold just 1–3 units. This shows there might be weak demand or not much market visibility, instead of being affected by pricing problems.
- Since several products sell in low numbers, businesses need to think about factors like seasonality, price comparison, and availability in stores to make better choices about keeping or removing them.

Overall, these results show that low-sales SKUs require careful evaluation instead of being removed. Businesses can attempt to use discounts or create promotional bundles to increase sales for items that sell the least. They should also assess how seasonal items perform after busy periods, such as the Climate Control season and keep an eye on sales trends in the coming months. This method improves the way SKUs are managed and boosts the performance of the entire category.

RQ13 - Top Products Generating After-Sales Service Demand

Which five products receive the highest number of service requests?

Purpose

This research question aims to find out which products cause the most work in after-sales service. By listing products based on how many service requests they get, the company can focus on improving quality, planning recalls or inspections, managing spare parts, and adjusting warranty policies.

Data Structure Consideration

Each service request links to a product using its product ID. Since several requests might come from the same product, you need to group them by product. To analyse recent patterns, you can limit it to certain dates and remove duplicate requests for the same issue if necessary.

Involved Tables

- Service_Request
- Product

Analytical Approach

Count how many service requests each product has, then rank them from highest to lowest. Show the top 5 products along with their brand and model details for easy understanding.

SQL Query

```
SELECT
  p.productid,
  p.brand,
  p.model,
  COUNT(sr.serviceid) AS service_count
FROM service_request sr
JOIN product p ON p.productid = sr.productid
-- Optional recency filter:
-- WHERE sr.requestdate BETWEEN :start_date AND :end_date
GROUP BY p.productid, p.brand, p.model
ORDER BY service_count DESC, p.productid
LIMIT 5;
```

Query Output

*Table 13 lists the **top 5 products** by service request volume, including product ID, brand, model, and the total number of requests (service_count).*

	productid [PK] integer	brand character varying (50)	model character varying (50)	service_count bigint
1	26	Samsung	EcoWash 10kg	2
2	30	Dyson	V11 Absolute	2
3	28	Beko	CoolSmart 12000BTU	2
4	4	Vestel	CoolPro 18000BTU	2
5	22	Philips	HairDry 2200	2

Results & Interpretation

The query results show the top products creating the most demand for after-sales services. These include Samsung EcoWash 10kg, Dyson V11 Absolute, Beko CoolSmart 12000BTU, Vestel CoolPro 18000BTU, and Philips HairDry 2200.

The findings reveal that:

- Each product listed had the same number of service requests (2 each). This created a tie for the top spot, showing a shared heavy demand for service instead of one main problem standing out.
- The range of products — including washing machines, cordless vacuums, air conditioners, and hair dryers — shows that service needs are spread across different categories rather than being focused on just one type of product.

These results show that after-sales service demands are shared among various product categories, so companies need to distribute resources. Companies need to check and prepare spare parts for these products. They should also match service requests with sales data to figure out how reliable the products are and find underlying issues like poor installation, bad usage, or faulty parts. Watching service patterns based on month and location can help spot early signs of problems and make it easier to improve product quality in advance.

RQ14 - Time-to-Failure Proxy for Major Home Appliances (White Goods)

*For white goods, how many days after a product's **first recorded sale** is a service request typically opened?*

Purpose

Calculate early reliability by counting the number of days from when the product is first sold to when the first service request comes in. This simple method points out early-life problems or failures and helps shape warranty rules, user guidelines, and planning for spare parts

Data Structure Consideration

Sales dates for products come from the Orders-Order_Item relationship, where using MIN(orderdate) reveals when a product was sold for the first time. The Service_Request table keeps track of issues that happen after a product is sold, and the Category table links products to white-goods. Since a single product can have more than one service ticket, joining these tables at the row level leads to repeated records. To calculate a *typical* time-to-failure, find the median or average time either by product or category (refer to the aggregate query). Leave out Cancelled and Returned orders when figuring out the first sale date. Make sure that requestdate is on or after the first_sale_date to calculate valid differences.

Involved Tables

- Orders
- Order_Item
- Service_Request
- Product, Category

Analytical Approach

The analysis looks at the number of days between the first recorded sale and the first service request for each white-good product. It uses "requestdate – first_sale_date" as a way to estimate the time until a failure happens. The results go through filtering based on important categories, and you can group them by product or category to find either the median or average reliability times.

SQL Query

```
SELECT
  c.categoryname,
  p.productid,
  p.brand,
  p.model,
  fs.first_sale_date,
  sr.requestdate,
  (sr.requestdate - fs.first_sale_date) AS days_after_first_sale
FROM service_request sr
JOIN product  p ON p.productid  = sr.productid
JOIN category c ON c.categoryid = p.categoryid
JOIN (
  SELECT
    oi.productid,
    MIN(o.orderdate) AS first_sale_date
  FROM orders o
  JOIN order_item oi ON oi.orderid = o.orderid
  WHERE o.status NOT IN ('Cancelled','Returned')
  GROUP BY oi.productid
) fs ON fs.productid = sr.productid
WHERE c.categoryname IN (
```

```
  'Refrigerators','Freezers','Dishwashers','Ovens','Microwaves',
  'Cooktops','Washing Machines','Dryers','Water Dispensers'
)
  AND sr.requestdate >= fs.first_sale_date
ORDER BY days_after_first_sale DESC;
```

Query Output

Table 14 (row-level view) lists white-goods service tickets with: category, product ID/brand/model, firstSaleDate, requestDate, and days_after_first_sale.

	categoryname character varying (50)	productid integer	brand character varying (50)	model character varying (50)	first_sale_date date	requestdate date	days_after_first_sale integer
1	Washing Machines	26	Samsung	EcoWash 10kg	2025-01-31	2025-04-26	85
2	Washing Machines	2	Arcelik	AquaWash 9kg	2025-01-07	2025-04-02	85
3	Microwaves	8	LG	QuickWave 25L	2025-01-13	2025-04-08	85
4	Dryers	18	Beko	DryPlus 8kg	2025-01-23	2025-04-18	85
5	Cooktops	20	Bosch	GasCook 4	2025-01-25	2025-04-20	85
6	Washing Machines	2	Arcelik	AquaWash 9kg	2025-01-07	2025-03-03	55
7	Microwaves	8	LG	QuickWave 25L	2025-01-13	2025-03-09	55
8	Dryers	18	Beko	DryPlus 8kg	2025-01-23	2025-03-19	55
9	Cooktops	20	Bosch	GasCook 4	2025-01-25	2025-03-21	55
10	Washing Machines	26	Samsung	EcoWash 10kg	2025-01-31	2025-03-27	55

Results & Interpretation

The query results highlight the top products that lead to the most demand for after-sales service. These include the Samsung EcoWash 10kg, Dyson V11 Absolute, Beko CoolSmart 12000BTU, Vestel CoolPro 18000BTU, and Philips HairDry 2200.

The findings show that:

- Each product had the same number of service requests, with 2 for each. This created a tie at the highest position, showing a shared heavy service demand instead of one standout issue.
- The different products involved, like washing machines, cordless vacuums, air conditioners, and hair dryers, show that after-sales needs spread across various product types instead of focusing on just one area.

The findings highlight how service workload spreads among various product groups making it crucial to create balanced operational plans. Companies need to focus on preparing spare parts and conducting technical inspections for these models. They should also match service requests with sales figures to determine service rates and do root-cause analysis, like checking installation quality, component issues, or user usage. Tracking monthly and regional service requests helps spot new reliability problems early and aids efforts to prevent issues and improve quality.

RQ15 - Best-Selling Products by Top-Level Category

For each top-level product category, which product achieves the highest sales volume (units sold), indicating the strongest demand within that category?

Purpose

Find the top-performing products in each main category to decide what to prioritize in assortments, promotions, and stock allocation. This helps identify key products even if total sales vary between categories.

Data Structure Consideration

Each product belongs to a specific category, and some categories may have parent categories. To group them under a top-level category, use COALESCE(parent.categoryname category.categoryname). Sales volume comes from Order_Item.quantity. Exclude orders that were canceled or returned to show actual demand.

Involved Tables

- Product
- Category (self-join as parent)
- Order_Item
- Orders (for status filtering)

Analytical Approach

Add up the total units sold for every product within each top-level category, then rank the products from highest to lowest by the number of units sold. Show products ranked as number one. If there's a tie for the top spot, display all the tied products as co-leaders.

SQL Query

```
WITH CategorySales AS (
  SELECT
    COALESCE(pc.categoryname, c.categoryname) AS top_category,
    p.productid,
    p.brand,
    p.model,
    SUM(oi.quantity) AS units_sold,
    RANK() OVER (
      PARTITION BY COALESCE(pc.categoryname, c.categoryname)
      ORDER BY SUM(oi.quantity) DESC
    ) AS sales_rank
  FROM orders o
  JOIN order_item oi ON oi.orderid = o.orderid
  JOIN product p      ON p.productid = oi.productid
  JOIN category c     ON c.categoryid = p.categoryid
  LEFT JOIN category pc ON pc.categoryid = c.parentcategoryid
  WHERE o.status NOT IN ('Cancelled','Returned')    -- realized demand
```

```
  GROUP BY COALESCE(pc.categoryname, c.categoryname),
           p.productid, p.brand, p.model
)
SELECT
  top_category,
  productid,
  brand,
  model,
  units_sold
FROM CategorySales
WHERE sales_rank = 1
ORDER BY top_category, brand, model;
```

Query Output

*Table 15 lists the **best-selling product(s)** per top-level category. In your sample, several categories have **ties** at the top (e.g., units_sold = 3).*

	top_category character varying (50)	productid [PK] integer	brand character varying (50)	model character varying (50)	units_sold bigint
1	Climate Control	14	Xiaomi	PureAir 4	3
2	Climate Control	4	Vestel	CoolPro 18000BTU	3
3	Entertainment	29	Philips	Ambilight 50	3
4	Kitchen Appliances	7	Siemens	BakeMaster 700	2
5	Kitchen Appliances	20	Bosch	GasCook 4	2
6	Kitchen Appliances	27	Vestel	CleanPro 12	2
7	Laundry Appliances	2	Arcelik	AquaWash 9kg	2
8	Major Appliances	19	Arcelik	DeepFreeze 300	3
9	Small Appliances	24	Arcelik	ToastPro XL	3
10	Small Appliances	9	Delonghi	Barista Plus	3

Results & Interpretation

The query results show 10 rows, each highlighting the best-selling product or products in a top-level category. Since RANK() determines the order, tied categories can have more than one leader.

Here are the findings:

- For Climate Control, Xiaomi PureAir 4 and Vestel CoolPro 18000BTU share the lead, both selling 3 units.
- To entertain, Philips Ambilight 50 takes the top spot with 3 units sold.
- To use in the kitchen, Siemens BakeMaster 700, Bosch GasCook 4, and Vestel CleanPro 12 all tie for the most sales, with 2 units each.
- To do laundry, Arçelik AquaWash 9kg secures the lead by selling 2 units.
- Major Appliances: The Arçelik DeepFreeze 300 takes the lead with 3 units sold.

- Small Appliances: Both the Arçelik ToastPro XL and Delonghi Barista Plus share the top spot, each with 3 units sold.

These results show the best-selling items in each category, which can guide product selection and stock management. When there are ties, consider using metrics like profit margins, service rate (requests versus units sold), or seasonality and placement tests to decide, or keep multiple leaders if space allows.

8. Web Dashboard

8.1 Purpose

This section documents the web dashboard built to visualize the same relational database used in SQL analyses. It transforms RQ1–RQ15 analytical outputs into actionable KPIs for non-technical users, providing a visual and interactive interface for sales, stock, and after-sales performance tracking.

8.2 Tech Stack & Data Binding

Frontend: React (Vite + TypeScript, dark theme UI)

Backend: Node.js / Express (API endpoints for Orders, Payments, Service Requests)

Database: PostgreSQL (normalized relational schema)

Styling: TailwindCSS, Chart.js (for KPI charts)

Data Source: Orders, Order_Item, Delivery, Payment, Product, Category, Inventory, Service_Request, Location, Store

Binding:

Dashboard Element	Data Source (Table)	Binding / SQL Field	Description
Total Orders	`Orders`	`COUNT(OrderID)`	Counts all sales orders in the system.
Total Revenue	`Orders`	`SUM(totalAmount)` *(excluding Cancelled/Returned)*	Represents total monetary value of valid sales.
Delivered Orders	`Orders`	`WHERE Status = 'Delivered'`	Number of orders successfully completed and received by customers.
In-Transit Deliveries	`Delivery`	`WHERE Status = 'In Transit'`	Active deliveries currently being shipped to customers.

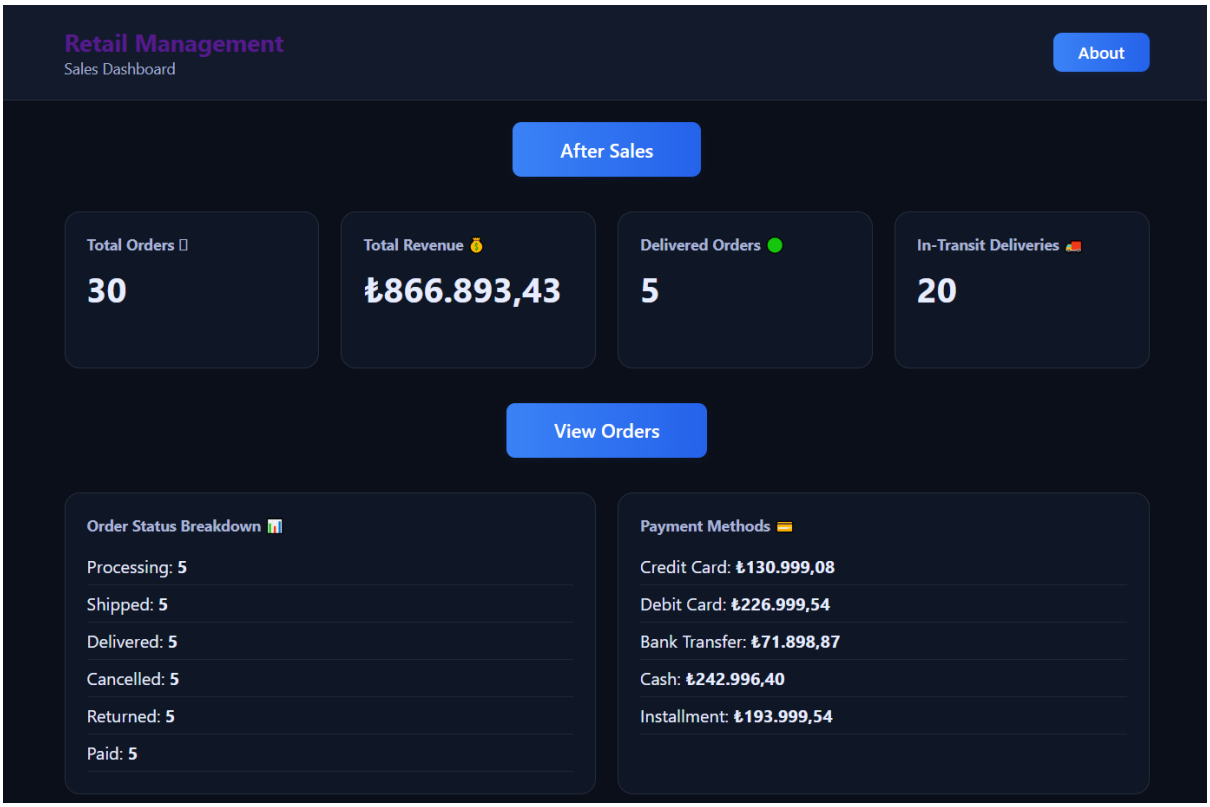
Order Status Breakdown	`Orders`	`GROUP BY Status`	Summarizes counts of orders under each operational status (Processing, Shipped, Returned, etc.).
Payment Methods	`Payment`	`GROUP BY PaymentMethod → SUM(Amount)`	Distribution of total revenue by payment channel (Credit, Debit, Cash, Bank Transfer, Installment).
Product Stock Overview	`Inventory + Product + Location`	`SELECT ProductName, Quantity, City, Region`	Displays stock levels of each product across regional stores.
Low Stock Alerts	`Inventory`	`WHERE Quantity < 85`	Highlights items below critical stock threshold (configurable KPI from RQ10).
Service Requests (Summary)	`Service_Request`	`COUNT(ServiceID)`	Total after-sales service tickets recorded in the system.
Service Requests by Product	`Service_Request + Product`	`GROUP BY ProductID → COUNT(ServiceID)`	Identifies top-performing or problematic products by number of service calls.
Orders Table (List)	`Orders + Customer`	`SELECT OrderID, CustomerName, Date, Status, TotalAmount`	Row-level display of order details for traceability (RQ3, RQ4).
Service Request Table (List)	`Service_Request + Customer + Product`	`SELECT ServiceID, CustomerName, ProductName, Date, Status`	Ticket-level after-sales detail supporting RQ11–RQ14.
ER Diagram (Schema)	—	—	Shows full entity relationships across Customer, Order, Product, Payment, and Service entities (*see Figure 9.7*).

Note:
All dashboard metrics are bound directly to the relational database schema used in the SQL analyses (see Figure 9.7). The KPIs correspond to queries presented in Section 8, ensuring analytical and visual consistency between the report and the dashboard.

8.3 Screens & Placement Guide

Figure 2 - Sales Dashboard Overview (KPIs)

Overview KPIs (Sales Dashboard}



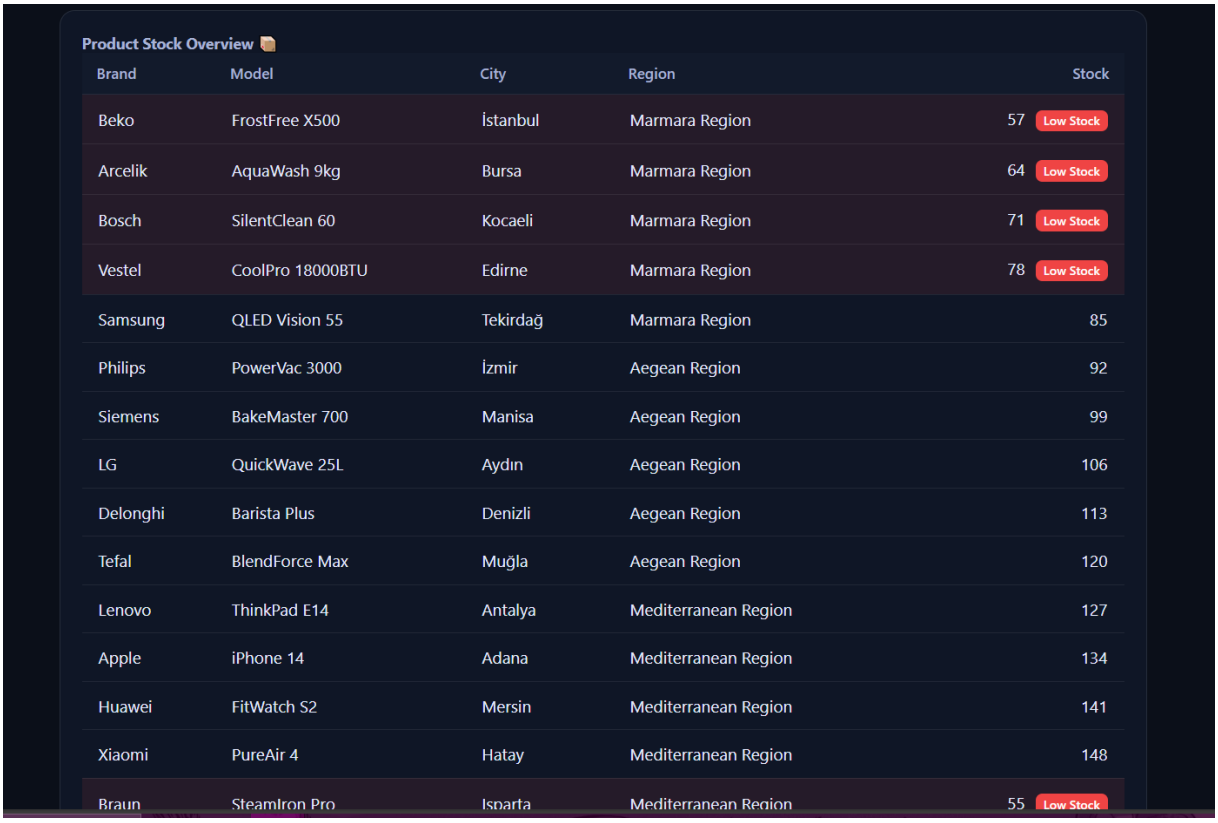
This screen displays Total Orders (30), Total Revenue (₺866,893.43), Delivered Orders (5), and In-Transit Deliveries (20), together with an Order Status Breakdown and Payment Methods summary.

It provides a high-level KPI view that links revenue, payment channel mix, and order flow to overall business performance.

This visualization supports the analytical results in RQ2 (average order value), RQ4 (order status mix), and RQ7 (payment–AOV relationship).

Figure 3 - Product Stock Overview with Low-Stock Flags

(Inventory Dashboard – Regional Stock Table)



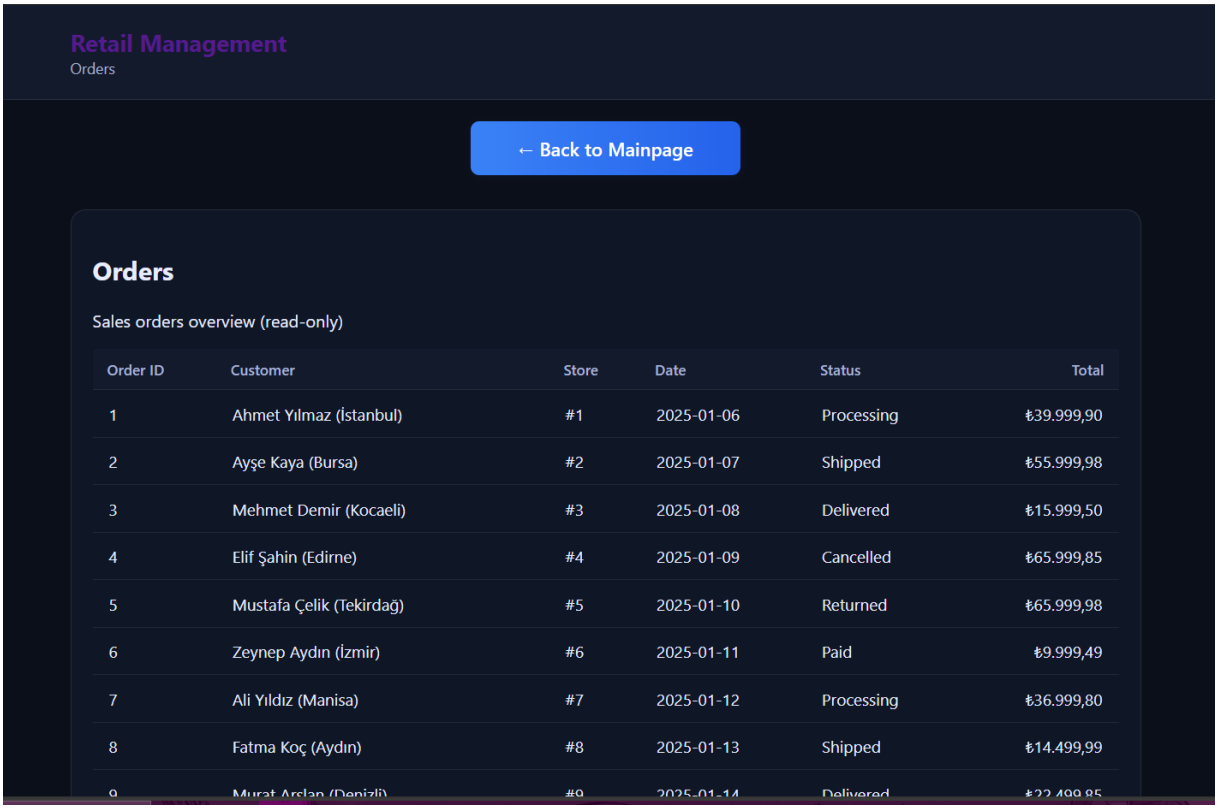
This screen presents a tabular inventory view by Brand / Model / City / Region / Stock. Rows below the configured 85-unit threshold are automatically flagged as *Low Stock*, helping identify potential shortages before they affect sales.

The visualization supports regional stock concentration analysis and connects directly to RQ9 (regional inventory patterns) and RQ10 (stock-threshold optimization).

Inventory table displaying brand, model, city, region, and stock quantity; rows below 85 units highlighted in red as Low Stock.

Figure 4 - Orders List

(Sales Dashboard – Transaction-Level Order Table)



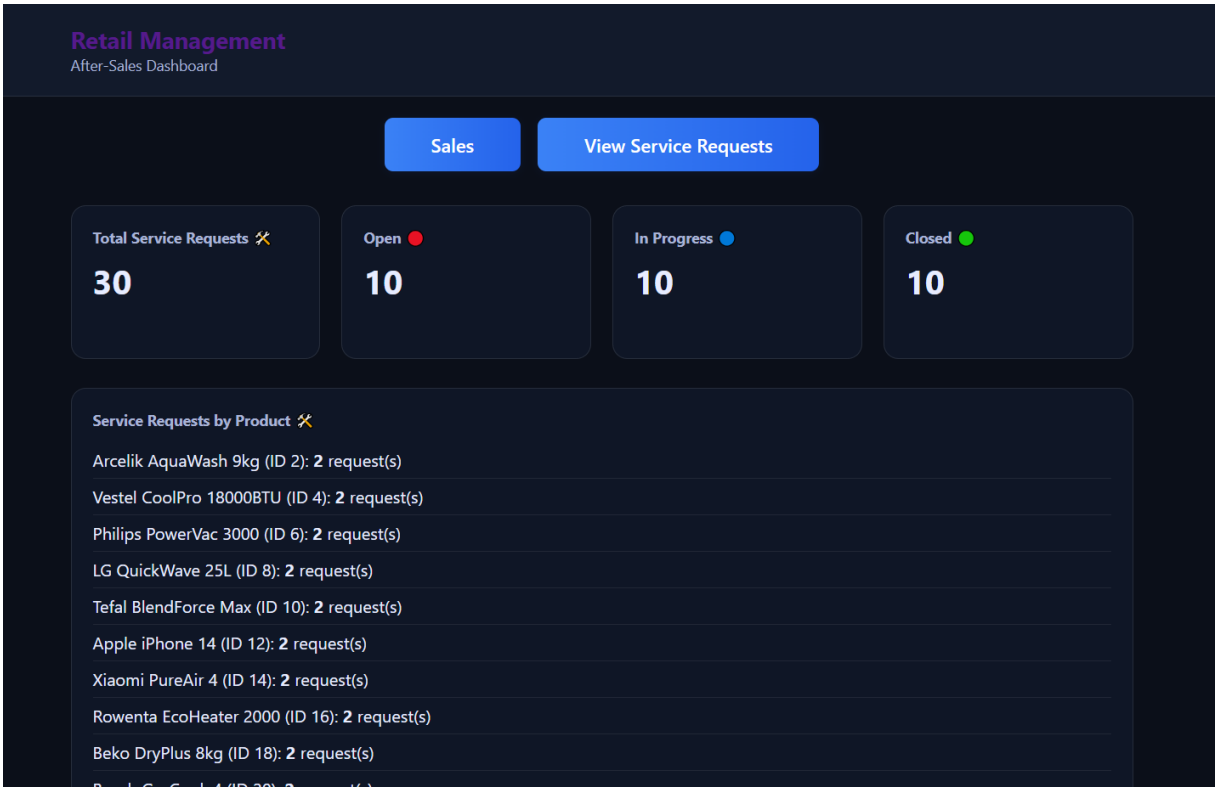
This view displays all sales orders in a transaction-level format, listing Order ID, Customer, Store, Date, Status, and Total Amount.

It provides visibility into store-level revenue distribution (RQ3) and order status traceability (RQ4), linking aggregate KPIs from the main dashboard back to individual order records. This screen ensures full transparency between high-level metrics and raw operational data.

Orders table displaying order ID, customer, date, status, and total amount columns for tracking transaction history.

Figure 5 - After-Sales Dashboard (Tickets by Status & Product)

(After-Sales Dashboard – Service Workload Summary View)



This dashboard summarizes overall service ticket volume and workload distribution across different status categories (Open, In Progress, Closed).

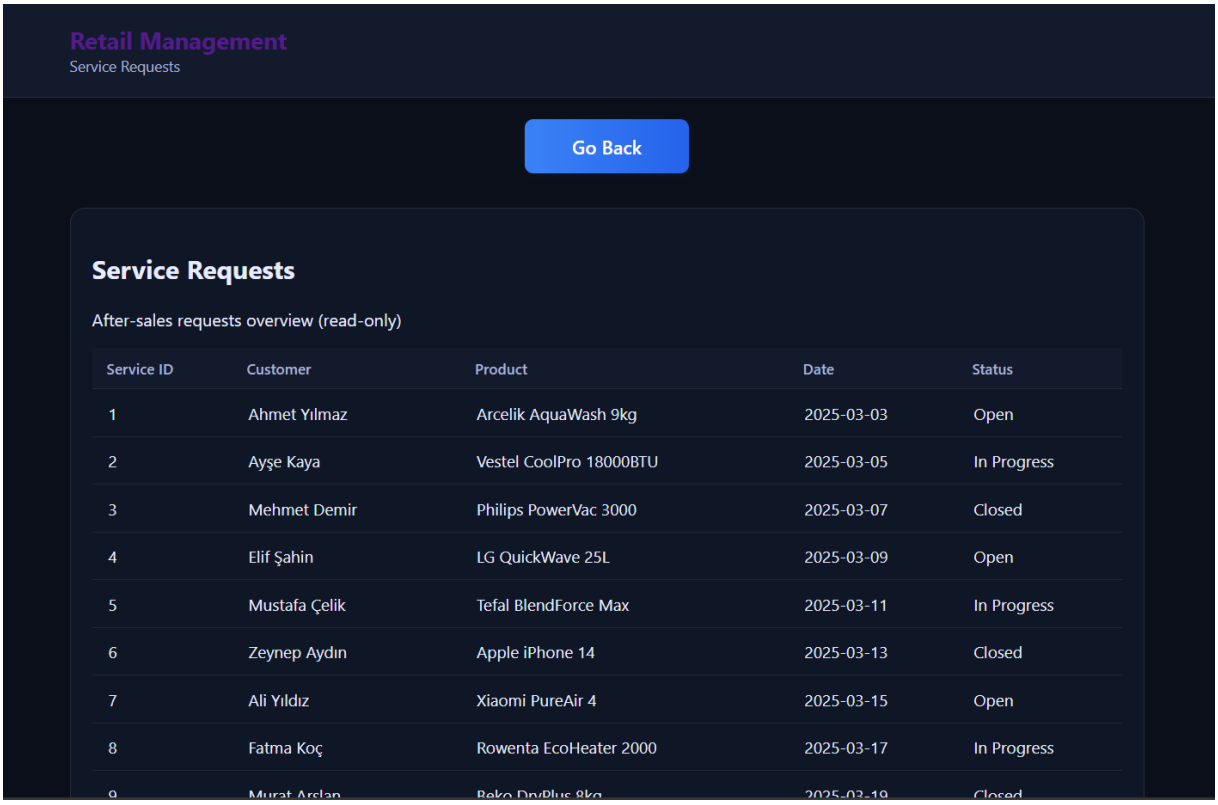
The product-level list highlights RQ13 (top products by service demand) and also supports RQ6 (category-level service concentration).

It provides managers with an operational view of post-sales performance and workload balance across all product segments.

After-sales dashboard displaying total service requests and their status (open, in progress, closed), plus a product-wise breakdown of request counts.

Figure 6 - Service Requests (Ticket-Level Detail)

(After-Sales Dashboard – Ticket-Level Service Data)



This detailed table presents ticket-level service request data, listing Service ID, Customer, Product, Date, and Status.

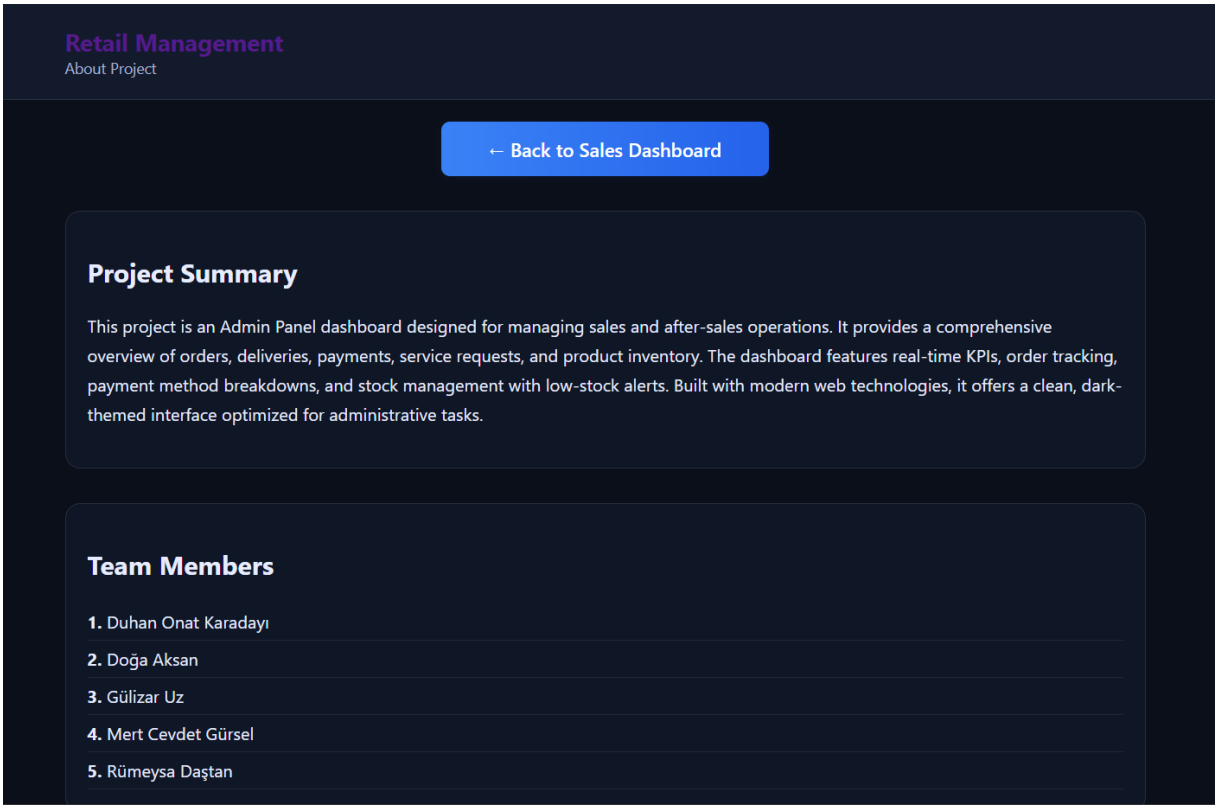
It enables proactive outreach (RQ11) by identifying customers linked to repeated or high-priority service requests, and supports reliability timing analysis (RQ14) by comparing request dates against first sale dates.

This screen ensures traceability between customer incidents and product performance, validating operational handling across regions and categories.

Service requests table showing service ID, customer, product, request date, and status for tracking after-sales operations.

Figure 7 - About Project (Summary & Team)

(Informational Page – Context for Reviewers & Contributors)



This page presents the Project Summary and Team Members, outlining the dashboard's purpose, scope, and key contributors.

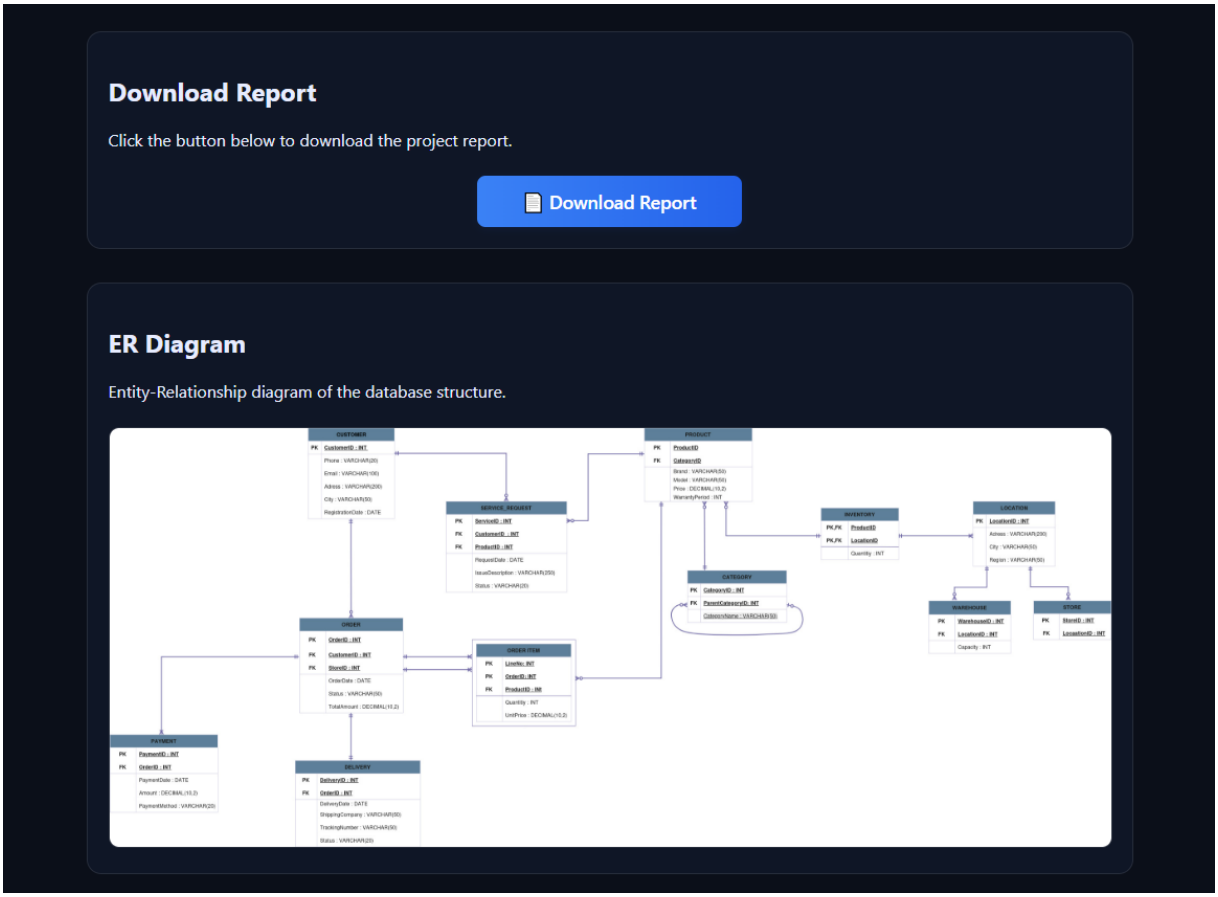
It provides reviewers with essential context for evaluation and clarifies each member's role during presentations or demonstrations.

The section highlights how the dashboard integrates analytical KPIs (from RQ1–RQ15) into a user-friendly interface built with modern web technologies.

"About Project" page displaying project summary paragraph and a list of team members with a dark-themed interface.

Figure 8 - Report Download & ER Diagram (Data Model Traceability)

(Schema Visualization & Report Export Section)



This section provides one-click access to download the project report and displays the Entity–Relationship (ER) Diagram representing the complete database schema.

The diagram aligns exactly with the SQL structure used in RQ1–RQ15, ensuring full traceability between the dashboard interface and the analytical queries.

It validates that every KPI shown in the web dashboard is derived directly from the same relational data model documented throughout the report.

ER diagram showing database tables and their relationships (Orders, Customers, Products, Payments, Deliveries, etc.), plus a download report button for exporting the project summary.

8.4 How the Dashboard Supports the RQs (Cross-Walk)

This section links each Research Question (RQ) to its matching visual in the Web Dashboard.

This helps keep a clear connection between SQL results and KPI views.

- **Sales & Market (RQ1–RQ5):** *Figure 9.1 (KPIs) + Figure 9.3 (Orders)*
  These visuals show data by category, region, and revenue distribution.
- **Operations & Inventory (RQ6–RQ10):** *Figure 9.1 (Status & Payments)*, *Figure 9.2 (Stock)*, *Figure 9.3 (Orders)*
  These figures highlight logistics, stock alerts, and delivery status.
- **Customer & Service (RQ11–RQ15):** *Figure 9.4 (After-Sales)*, *Figure 9.5 (Tickets)*, *Figure 9.7 (ER Model)*
  These visuals connect after-sales data with customer feedback and timing patterns.

8.5 Limitations & Next Steps

The prototype dashboard effectively displays key KPIs. However, there are several opportunities for improvement in future versions:

- Currently, the dashboard relies on a demo snapshot and does not support live data write-back.
- We should add **filters** for *date, region, and category, as well as export and authentication features.*
- Scheduling **ETL automation** will ensure KPIs refresh automatically. Enabling drill-down from category to product, order, and ticket will enhance usability.
- To improve reliability analytics, link **RQ14** to a *time-to-failure chart* that displays the median days to first ticket by category.

9. Results & Discussion

This section summarizes and interprets the key findings from the analysis of the previous two sections, 8 and 9. Analyzing each of the 15 research questions (RQ1 to RQ15) has contributed to a better understanding of the interlinkage of the sales, operations, and after-sales functions of the firm. The findings derived from the SQL were complemented with the dashboards, further strengthening the analytical linkage from data to decision.

The analysis brought to light a number of meaningful patterns and operational insights.

Demand at the Regional and Category Level

Sales analysis reveals a heavy concentration of sales activities in the Marmara and Ege regions. These two regions dominate sales of the firm for all other regions. In these two regions, sales of major appliances (e.g. refrigerators, and washing machines) and products related to climate control (e.g., air conditioners and heaters) dominate the sales portfolio. This is an indication of a regional inclination towards the acquisition of durable home goods and devices for temperature control, which is possibly related to seasonality and demography.

Operational and Payment Analytics

An analysis of the records of delivery and payment shows that credit and installment payment methods are associated with higher order values (AOV) as compared to cash or bank transfer payment methods. Moreover, the extent of delivery also varied among the regions, with some couriers completing the orders faster, but with limited coverage in the regions. These findings

10. Conclusion & Future Work

This project built a complete system that links SQL-based data exploration with a user-friendly React dashboard. The platform shows important business processes and maintains traceability from raw data structures to important managerial insights. By organizing relational databases well, the system proved they can handle both descriptive and diagnostic analytics in real-world retail settings.

Answering each research question highlighted patterns in operations. These patterns help businesses track sales, manage inventory, and measure the success of after-sales services. Turning static SQL queries into interactive dashboards made it easier for technical teams and decision-makers to work together. This approach encourages clearer communication and decisions based on data.

Future improvements might enhance how the system handles analytics and increase its automation. Setting up an ETL pipeline could keep metrics updated in real time without requiring people to update them. Advanced filtering and user login options could give secure access based on user roles. Predictive modeling has the potential to increase the system's intelligence by helping forecast demand, organize customers, and estimate service failure risks. Including features like hierarchical drill-downs from categories to products to orders to tickets, as well as making it mobile-friendly, could make the system more user-friendly.

Focusing on insights around reliability, like time-to-failure data, the project can move toward tracking maintenance ahead of time and enhancing performance.

To wrap up, the system achieved its main goal of creating a reliable and trackable analytical setup. It also built a solid base to scale into an automated and smart business intelligence solution. This solution helps make ongoing decisions based on data.

11. Authors' Contributions

The construction of this project involved equal and joint efforts from all members of the team.

The authors collaborated for the construction of the database, the preparation of the data, the SQL-based data analysis, and the elaboration of the results.

Everyone worked together on the following:

Designing the ER blueprint and the logical structure,

PostgreSQL table construction and data insertion,

Research Questions formulation and analysis (RQ1–RQ15),

SQL query construction and analytical output interpretation,

Web dashboard/application design and documentation,

Executive summation and presentation aid preparation.

Everyone participated in all phases of the project equally, and all members of the team contributed equally to the accomplishment of the project.

12. Appendix – SQL and Schema Reference Guide

This section gives an overview of the main technical elements that help with the analyses found in Sections 4 to 8 of this report. It works as a quick guide to understand the database structure how data was generated, and the SQL approach used during the research.

12.1.Understanding the Schema

Section 5.3 introduces the relational schema that focuses on the primary entities involved in studying sales, stock, and after-sales services. Each entity has a unique primary key, while foreign keys tie the entities to one another.

Key tables include:

- The Customer table stores personal information and contact details of individuals.
- The Orders and Order_Item tables record details of purchases made by customers and the products bought.
- The Product and Category tables show information about items and how they are grouped.
- Inventory and Location are used to monitor stock quantities in warehouses and stores across areas.
- Payment and Delivery link customer orders with payment processes and shipping details.
- The Service_Request table keeps logs of post-sale issues to analyze reliability and assess satisfaction levels.

This framework helps link daily operations to analyze and reach every research goal.

12.2 SQL Methodology

Researchers used structured SQL queries to address each research question (RQ1–RQ15) by running them on the relational database to gather and analyze data. Some examples of these analyses are:

- Product and Revenue Ranking focused on grouping order items by brand and model to find total revenue and sales.

- Category-Level Service Load examined how many service requests came from each category to identify where maintenance or quality problems were more frequent.
- Payment Method Comparison involved looking at average order values for each payment type to see how customer behaviors varied.
- Delivery Speed Evaluation checked how long deliveries took across different shipping companies to compare their performance in logistics.
- Regional Inventory Monitoring calculated stock levels in various regions to find any potential shortages or uneven distribution.

SQL queries followed relational integrity rules and left out cancelled or returned transactions to keep the data accurate.

12.3 Getting Data Ready

As explained in Section 5, the datasets were created using synthetic methods to mimic real retail operations based on the planned schema. The process started with inputting master data like customers, products, categories, and locations. after that came the transactional data like orders, payments, and deliveries. This order was necessary to keep foreign key links intact. By following this order, every analysis query gives valid and repeatable outcomes.

12.4 Traceability Note

Section 7 connects each SQL query used in the analysis to its corresponding Research Question (RQ). This clear link helps provide complete transparency between how the data is structured how the queries are written, and the results shown.