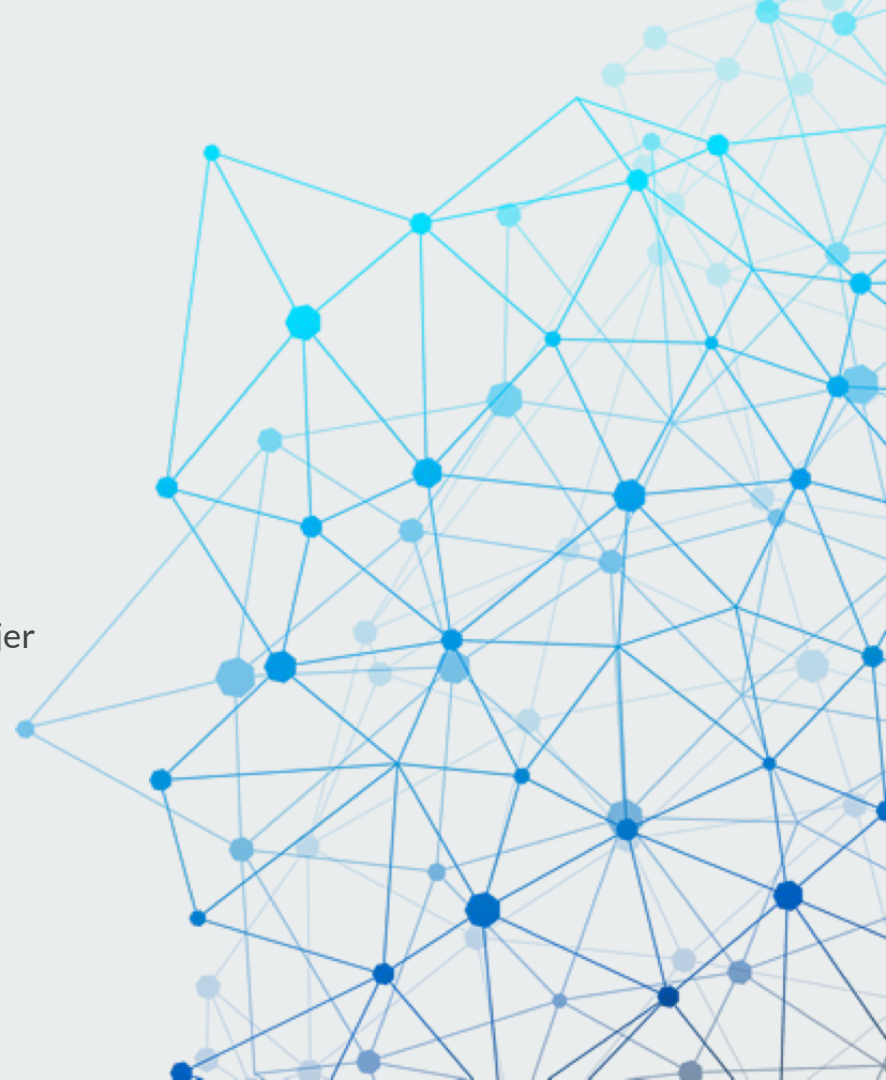
Retail Store Management System Database

Amir Jannatkhah, Bruno Shinji Akune, Cassandra Lemdjo, Hadjer Benbouzid, Rahul Balain

Course: Database Management System



Project Overview

Objective: Design a relational database for a retail store

Purpose: Manage products, customers, employees, suppliers, and receipts efficiently

Tools Used: SQL, ERD modeling, normalization techniques

Business Scenario

Type of Business: Retail store (general goods including cosmetics, food, furniture, cleaning supplies, and baby products)

Key Functionalities:

- Track inventory
- Manage customers, orders, and employees
- Store supplier and shipping information

Entity-Relationship Diagram (ERD)

Cardinalities:

Customer→Receipts: 0 to many

Employee→Receipts: 0 to many

Receipt→Receipt_items: 1 to Many

Product→Receipt_items: 1 to Many

Products→Category: Many to 1

Products→Shipper: Many to 1

Customer←Receipts: 1 to many

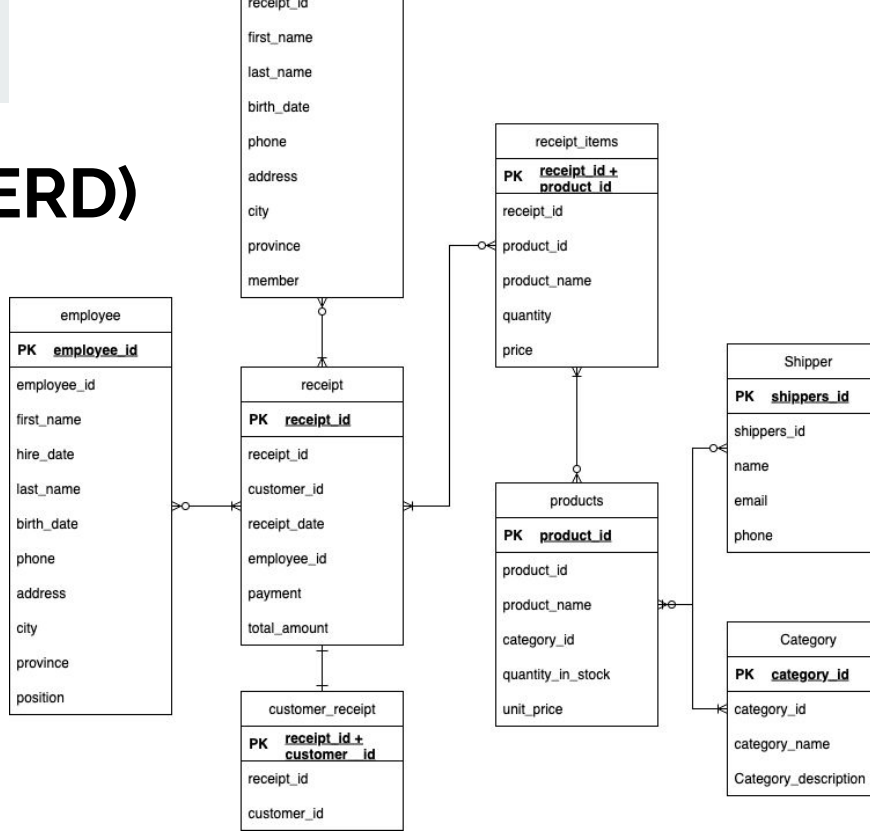
Employee←Receipts: 1 to many

Receipt←Receipt_items: 0 to many

Product←Receipt_items: 0 to many

Products←Category: 1 to many

Products←Shipper: 0 to many



Normalization Summary

Followed 1NF → 2NF → 3NF

Key goals of each normalization process:

- **1F**: Ensure all the columns have one value in a cell.
- **2F:** All the cells in the column depend on the primary key.
- **3F:** Separate any cells that aren't dependent on the primary, by creating another table and connecting them through a foreign key.

Normalization 1F - Example (Customers Table)

Before:

customer_id	receipt_id	first_name	last_name	birth_date	phone	address	city	province
1	1,2,3	Arnold	Smith	1990-02-22	289-956-9091	111 5th Avenue St	Toronto	ON

Normalization 1F - Example (Customers Table)

After:

customer_id	receipt_id	first_name	last_name	birth_date	phone	address	city	province
1	1	Arnold	Smith	1990-02-22	289-956-9091	111 5th Avenue St	Toronto	ON
1	2	Arnold	Smith	1990-02-22	289-956-9091	111 5th Avenue St	Toronto	ON
1	3	Arnold	Smith	1990-02-22	289-956-9091	111 5th Avenue St	Toronto	ON

Later we would remove the receipt_id column. To accommodate the 2F-3F normalization.

Customer_receipts

Receipt_id	customer_id

A similar process was made with the receipt table, because we had product_id in it, and since one receipt can have many products, we had to rework the column. As a result, we created the Receipt_items, where it will expand on each item in a receipt.

Normalization 2F- Example 2 (Receipts Table)

Before:

receipt_id	customer_id	product_id	receipt_date	employee_id	payment	total_amount
1	1	21	2025-05-01	101	Credit	110.20
1	1	30	2025-05-01	101	Credit	10.20
1	1	40	2025-05-01	101	Credit	15.05

After:

receipt_id	customer_id	receipt_date	employee_id	payment	total_amount
1	1	2025-05-01	101	Credit	110.20

We removed the product_id column because in the receipt table it didn't depend on the receipt_id. Therefore it doesn't need to be in this table, but in another. **We created a table only for items in a receipt.**

Receipts_items

receipt_id	product_id	product_name	quantity	total_price

Normalization 3f - Example 3

When creating the receipt_items table, we also adapted it to only display the products on that receipt. For a more detailed table about the product we adapted the product table to store the data about each product. This ensures that table keys will depend on their own primary key without having any additional dependencies or multiple values .

receipt_id	customer_id	receipt_date	employee_id	payment	total_amount

Receipt table

receipt_id	product_id	product_name	quantity	total_price

Receipts_items table

product_id	product_name	category_id	quantity_in_stock	unit_price	shippers_id

Products table

SQL Implementation Highlights

Database Schema Creation

Our schema was built based on the normalized ERD, translating entities into SQL **CREATE TABLE** statements. Each table was carefully structured with:

- **Primary Keys** – to uniquely identify each record.
- **Foreign Keys** – to enforce referential integrity across tables.
- **Data Types** – chosen based on the nature of each attribute (e.g., VARCHAR for names, DATE for dates, DECIMAL for prices).
- **Constraints** – to maintain data quality.

SQL Implementation Highlights

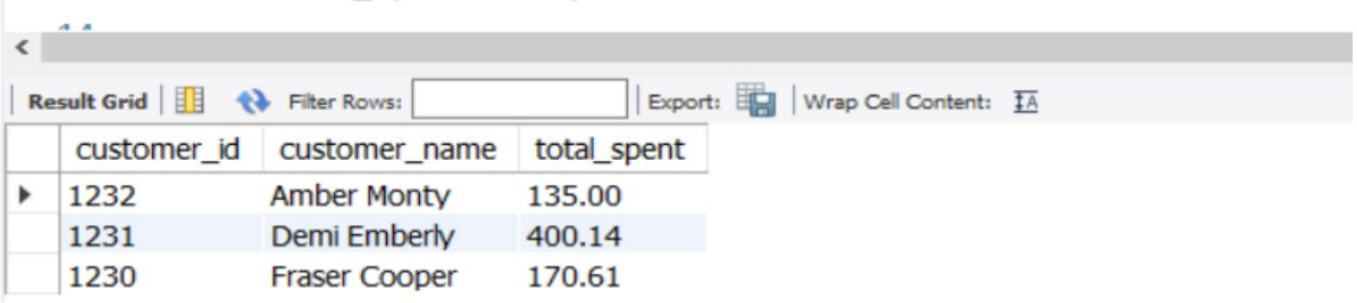
Feature	Example	Purpose
PRIMARY KEY	PRIMARY KEY (product_id)	Uniquely identifies each product
FOREIGN KEY	FOREIGN KEY (category_id) REFERENCES Category(category_id)	Links products to their categories
NOT NULL	first_name VARCHAR(50) NOT NULL	Ensures critical data is always present
UNIQUE	email VARCHAR(100) UNIQUE	Prevents duplicate emails (e.g., shippers)
COMPOSITE KEY	PRIMARY KEY (receipt_id, product_id) in Receipt_items	Handles many-to-many relationships

SQL Query 1 – Total Sales by Customer

```sql
# Which customers have generated the most revenue for the business?

SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    SUM(r.total_amount) AS total_spent
FROM
    Customers c
JOIN
    Receipts r ON c.customer_id = r.customer_id
GROUP BY
    c.customer_id
ORDER BY
    total_spent DESC;
```

Understanding customer spending patterns is essential for marketing strategies, loyalty programs, and customer relationship management.

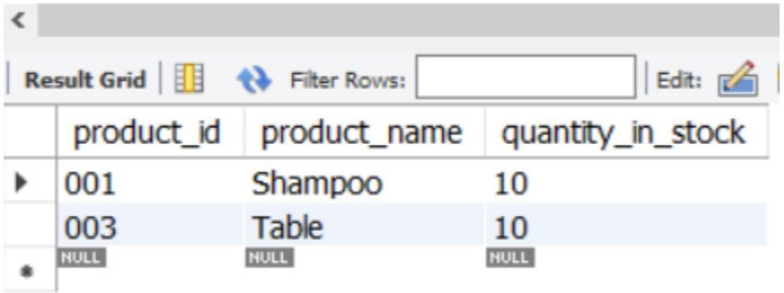


customer_id	customer_name	total_spent
1232	Amber Monty	135.00
1231	Demi Emberly	400.14
1230	Fraser Cooper	170.61

SQL Query 2 – Low Stock Products

```sql
# Which products are low in stock and may need restocking?

SELECT
    product_id,
    product_name,
    quantity_in_stock
FROM
    Products
WHERE
    quantity_in_stock < 20
ORDER BY
    quantity_in_stock ASC;
```



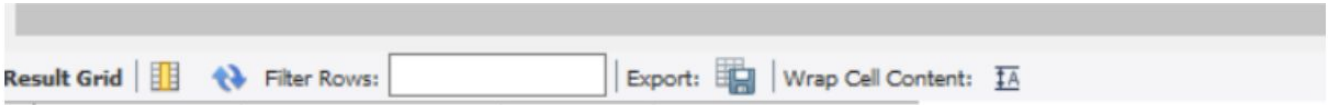
product_id	product_name	quantity_in_stock
001	Shampoo	10
003	Table	10
NULL	NULL	NULL

It's a crucial part of inventory management, helping the business avoid stockouts and maintain customer satisfaction.

SQL Query 3 – Employee Roles & Activity

```sql
# What roles do employees have, and how many receipts have they handled?

SELECT
    e.employee_id,
    CONCAT(e.first_name, ' ', e.last_name) AS employee_name,
    e.position,
    COUNT(r.receipt_id) AS receipts_handled
FROM
    Employees e
LEFT JOIN
    Receipts r ON e.employee_id = r.employee_id
GROUP BY
    e.employee_id
ORDER BY
    receipts_handled DESC;
```



employee_id	employee_name	position	receipts_handled
202	Anthony Kids	Cashier	7
203	Sabrina Jeans	Cashier	3
201	John Smith	Manager	0

This gives us insight into employee performance.

SQL Query 4 –Revenue Breakdown by Top-Selling Items

```sql
-- Which product categories generate the most revenue, and what are the top-selling items in each category?
-- What are the suppliers for those top-selling products in each category?

SELECT
    cat.category_name,
    p.product_name,
    SUM(ri.total_price) AS total_sales,
    s.name AS supplier_name,
    s.email AS supplier_email,
    s.phone AS supplier_phone
FROM
    Receipt_items ri
JOIN
    Products p ON ri.product_id = p.product_id
JOIN
    Category cat ON p.category_id = cat.category_id
JOIN
    Shippers s ON p.shippers_id = s.shippers_id
GROUP BY
    cat.category_name, p.product_name, s.name, s.email, s.phone
ORDER BY
    cat.category_name, total_sales DESC;
```

help the business make smarter decisions by revealing key insights about inventory and product profitability.

category_name	product_name	total_sales	supplier_name	supplier_email	supplier_phone
Baby product	Diapers	377.55	Johnny's Shipping	jonny.shipping@shippers.ca	613-254-2315
Cleaning	Multi-surface cleaner	27.99	Truck	truck@shippers.com	434-256-2897
Cosmetics	Shampoo	75.00	Johnny's Shipping	jonny.shipping@shippers.ca	613-254-2315
Cosmetics	Conditioner	50.00	Johnny's Shipping	jonny.shipping@shippers.ca	613-254-2315
Cosmetics	Soap	25.00	Johnny's Shipping	jonny.shipping@shippers.ca	613-254-2315
Food	Cake	57.23	Express_shipping	express.shipping@shippers.ca	434-258-8018
Food	Popcorn	56.00	Express_shipping	express.shipping@shippers.ca	434-258-8018

Key Achievements !

- **Normalization to 3NF**:
  - Reduced redundancy and ensure data integrity
  - Created meaningful relationships between tables via foreign keys and linking tables
- **Efficient Schema Design**:
  - Structured all entities with proper constraints, data types, and indexing
  - Applied **composite keys** for managing many-to-many relationships (e.g., receipt items)
- **SQL Querying Power**:
  - Developed insightful queries to:
    - Track sales performance by customer or product
    - Detect low-stock inventory
    - Measure employee performance
    - Analyze revenue trends by product category

Conclusion & Project Summary!

We successfully designed and implemented a relational database system for a retail store that:

- Tracks customers, products, receipts, employees, shippers, and categories
- Handles inventory, transactions, and supplier logistics
- Provides insightful **SQL-driven analytics** for business decision-making