

EASTERN MEDITERRANEAN UNIVERSITY

Department of Computer Engineering



CMPE353 / CMSE351 / AING353

DATABASE MANAGEMENT SYSTEMS

LABS 2 - 7

Design and Implementation of a Multi-Branch Retail Management System

Fall 2026-2027

Platform	Oracle Live SQL
Database scope	12 relational tables
Main areas	Branches, products, inventory, purchasing, sales
Assessment period	Labs 2-7

This specification is the official semester database case study for the laboratory work.

1. Task Overview

You are asked to implement a relational database for a multi-branch retail business using Oracle Live SQL. The same database will be developed progressively throughout Labs 2-7. Each laboratory task builds on the work completed in the previous lab, so students must keep one consistent version of the database throughout the semester.

Important: The table names and column names in this document must be used exactly as specified so that lab demonstrations, query checking, and final assessment can be performed consistently.

2. Problem Statement

The system represents a retail company that operates multiple branches. Each branch employs staff, keeps local inventory, purchases products from suppliers, and records customer sales.

Products are organized into categories. Inventory is maintained separately for each branch because the same product may have different stock quantities at different locations. This creates a many-to-many relationship between branches and products that is resolved through the `branch_inventory` table.

Purchasing records describe stock obtained from suppliers. A purchase belongs to one branch and one supplier and contains one or more `purchase_items`. Sales record transactions made at branches. A sale is handled by an employee, may optionally be linked to a registered customer, and contains one or more `sale_items`.

The schema is intentionally limited to twelve tables so that students can focus on relational design, key constraints, joins, aggregation, nested queries, transactions, triggers, and different relationship cardinalities without unnecessary application-level complexity.

3. Core Business Rules

- A branch can employ many employees, but each employee is assigned to exactly one branch.
- Each employee has exactly one `employee_profile`, and each `employee_profile` belongs to exactly one employee. The `employee_profiles.employee_id` column is both the primary key and a foreign key to `employees.employee_id`, implementing the one-to-one relationship with a shared key.
- A category can contain many products, but each product belongs to exactly one category.
- A product can be stocked in many branches and a branch can stock many products. The `branch_inventory` table resolves this relationship.
- A supplier can have many purchase records. Each purchase is made for one branch and is recorded by one employee.
- A purchase must contain one or more `purchase_items`. Each purchase item refers to one product.
- A branch can have many sales. Each sale is handled by one employee who belongs to that branch.
- A sale may belong to a registered customer; `customer_id` is nullable to support walk-in customers.
- A sale must contain one or more `sale_items`. Each sale item refers to one product.
- Quantities, prices, costs, totals, loyalty points, and reorder levels must never be negative.
- Only the predefined status and payment values specified in the schema are valid.

4. Database Schema Summary

No.	Table	Purpose
1	<code>branches</code>	Stores physical branch information.
2	<code>employees</code>	Stores employees and their assigned branch.
3	<code>customers</code>	Stores registered customer information.

No.	Table	Purpose
4	suppliers	Stores supplier contact and status information.
5	categories	Classifies products into logical groups.
6	products	Stores the retail product catalog.
7	branch_inventory	Stores the quantity of each product in each branch.
8	purchases	Stores purchase headers for incoming stock.
9	purchase_items	Stores the products and quantities included in purchases.
10	sales	Stores sale headers for completed or cancelled transactions.
11	sale_items	Stores the products and quantities included in sales.
12	employee_profiles	Stores one personal profile for each employee (one-to-one).

5. Entity-Relationship Diagram

The diagram below shows the main primary-key and foreign-key relationships. Full attribute definitions appear in Section 6.

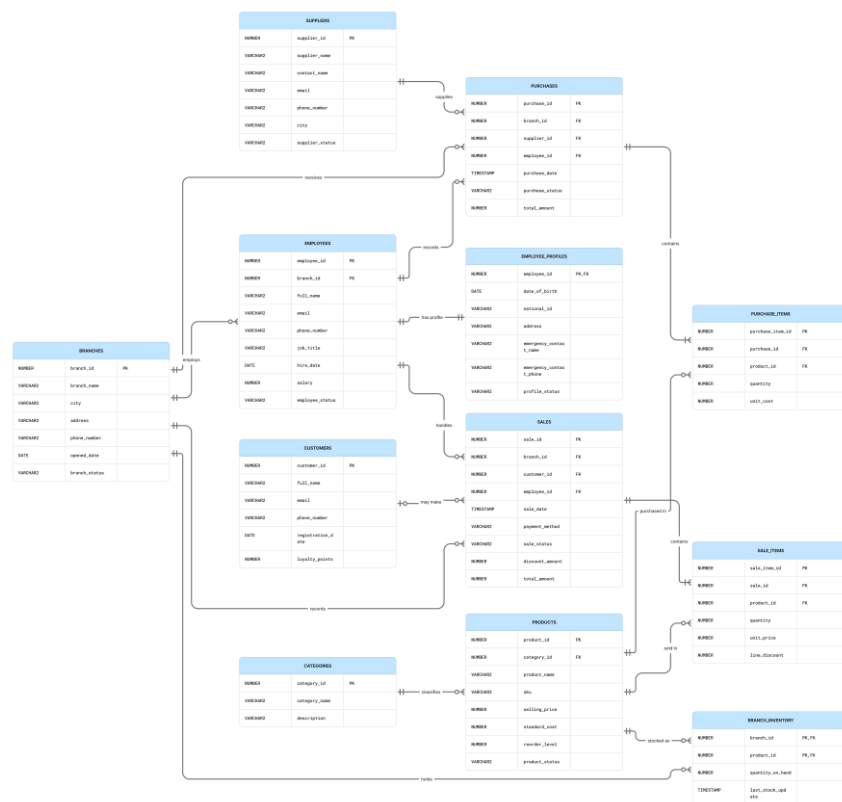


Figure 1: Entity-Relationship (ER) diagram of the Retail Management System.

For a clearer, full-size view of the diagram [open on Figma](#).

6. Detailed Table Definitions

Oracle convention: Use Oracle-compatible data types and constraints. Identity columns should be implemented using Oracle identity syntax. Students are expected to write the CREATE TABLE statements themselves from the specification below.

6.1 branches

Column	Data Type	Constraints	Description
branch_id	NUMBER	Primary Key; identity; Not Null	Unique identifier for a branch.
branch_name	VARCHAR2(100)	Unique; Not Null	Branch display name.
city	VARCHAR2(60)	Not Null	City in which the branch operates.
address	VARCHAR2(200)	Not Null	Physical branch address.
phone_number	VARCHAR2(25)	Unique; Not Null	Main branch contact number.
opened_date	DATE	Not Null	Date the branch started operating.
branch_status	VARCHAR2(15)	Default 'Active'; CHECK in ('Active','Inactive'); Not Null	Operational status of the branch.

6.2 employees

Column	Data Type	Constraints	Description
employee_id	NUMBER	Primary Key; identity; Not Null	Unique employee identifier.
branch_id	NUMBER	Foreign Key -> branches.branch_id; Not Null	Branch to which the employee is assigned.
full_name	VARCHAR2(120)	Not Null	Employee full name.
email	VARCHAR2(120)	Unique; Not Null	Employee email address.
phone_number	VARCHAR2(25)	Not Null	Employee contact number.
job_title	VARCHAR2(30)	CHECK in ('Manager','Cashier','StockClerk','Supervisor'); Not Null	Employee role in the branch.
hire_date	DATE	Not Null	Date employment began.
salary	NUMBER(10,2)	CHECK salary > 0; Not Null	Current monthly salary.
employee_status	VARCHAR2(15)	Default 'Active'; CHECK in ('Active','Inactive'); Not Null	Employment status.

6.3 customers

Column	Data Type	Constraints	Description
customer_id	NUMBER	Primary Key; identity; Not Null	Unique customer identifier.
full_name	VARCHAR2(120)	Not Null	Customer full name.
email	VARCHAR2(120)	Unique; Nullable	Customer email address, if provided.
phone_number	VARCHAR2(25)	Unique; Not Null	Customer contact number.
registration_date	DATE	Default SYSDATE; Not Null	Date the customer was registered.

Column	Data Type	Constraints	Description
loyalty_points	NUMBER(8)	Default 0; CHECK loyalty_points >= 0; Not Null	Current customer loyalty balance.

6.4 suppliers

Column	Data Type	Constraints	Description
supplier_id	NUMBER	Primary Key; identity; Not Null	Unique supplier identifier.
supplier_name	VARCHAR2(120)	Unique; Not Null	Registered or trading name of supplier.
contact_name	VARCHAR2(120)	Not Null	Main contact person.
email	VARCHAR2(120)	Nullable	Supplier email address.
phone_number	VARCHAR2(25)	Not Null	Supplier contact number.
city	VARCHAR2(60)	Not Null	Supplier city.
supplier_status	VARCHAR2(15)	Default 'Active'; CHECK in ('Active','Inactive'); Not Null	Whether the supplier is currently usable.

6.5 categories

Column	Data Type	Constraints	Description
category_id	NUMBER	Primary Key; identity; Not Null	Unique category identifier.
category_name	VARCHAR2(80)	Unique; Not Null	Category name such as Electronics or Groceries.
description	VARCHAR2(300)	Nullable	Optional description of the category.

6.6 products

Column	Data Type	Constraints	Description
product_id	NUMBER	Primary Key; identity; Not Null	Unique product identifier.
category_id	NUMBER	Foreign Key -> categories.category_id; Not Null	Category to which the product belongs.
product_name	VARCHAR2(140)	Not Null	Product display name.
sku	VARCHAR2(40)	Unique; Not Null	Stock keeping unit used to identify the product.
selling_price	NUMBER(10,2)	CHECK selling_price > 0; Not Null	Current retail selling price.
standard_cost	NUMBER(10,2)	CHECK standard_cost > 0; Not Null	Reference cost used by the business.
reorder_level	NUMBER(8)	Default 0; CHECK reorder_level >= 0; Not Null	Minimum desired stock level.
product_status	VARCHAR2(15)	Default 'Active'; CHECK in ('Active','Discontinued'); Not Null	Availability of the product for normal operations.

6.7 branch_inventory

Column	Data Type	Constraints	Description
branch_id	NUMBER	Primary Key (composite); Foreign Key -> branches.branch_id; Not Null	Branch holding the stock.
product_id	NUMBER	Primary Key (composite); Foreign Key -> products.product_id; Not Null	Product held by the branch.
quantity_on_hand	NUMBER(10)	Default 0; CHECK quantity_on_hand >= 0; Not Null	Current quantity available at the branch.
last_stock_update	TIMESTAMP	Default CURRENT_TIMESTAMP; Not Null	Timestamp of the latest inventory change.

6.8 purchases

Column	Data Type	Constraints	Description
purchase_id	NUMBER	Primary Key; identity; Not Null	Unique purchase identifier.
branch_id	NUMBER	Foreign Key -> branches.branch_id; Not Null	Branch receiving the purchase.
supplier_id	NUMBER	Foreign Key -> suppliers.supplier_id; Not Null	Supplier providing the goods.
employee_id	NUMBER	Foreign Key -> employees.employee_id; Not Null	Employee who records the purchase.
purchase_date	TIMESTAMP	Default CURRENT_TIMESTAMP; Not Null	Date and time of purchase record.
purchase_status	VARCHAR2(15)	Default 'Pending'; CHECK in ('Pending','Received','Cancelled'); Not Null	Current purchase status.
total_amount	NUMBER(12,2)	CHECK total_amount >= 0; Not Null	Recorded total purchase amount.

6.9 purchase_items

Column	Data Type	Constraints	Description
purchase_item_id	NUMBER	Primary Key; identity; Not Null	Unique purchase-line identifier.
purchase_id	NUMBER	Foreign Key -> purchases.purchase_id; Not Null	Purchase header to which the line belongs.
product_id	NUMBER	Foreign Key -> products.product_id; Not Null	Purchased product.
quantity	NUMBER(8)	CHECK quantity > 0; Not Null	Number of units purchased.
unit_cost	NUMBER(10,2)	CHECK unit_cost > 0; Not Null	Actual unit cost for this purchase.
(purchase_id, product_id)	-	Unique pair recommended	A product should normally appear once within a single purchase.

6.10 sales

Column	Data Type	Constraints	Description
sale_id	NUMBER	Primary Key; identity; Not Null	Unique sale identifier.
branch_id	NUMBER	Foreign Key -> branches.branch_id; Not Null	Branch at which the sale is recorded.

Column	Data Type	Constraints	Description
customer_id	NUMBER	Foreign Key -> customers.customer_id; Nullable	Registered customer; NULL means walk-in customer.
employee_id	NUMBER	Foreign Key -> employees.employee_id; Not Null	Employee handling the sale.
sale_date	TIMESTAMP	Default CURRENT_TIMESTAMP; Not Null	Date and time of the transaction.
payment_method	VARCHAR2(20)	CHECK in ('Cash','Card','BankTransfer','MobilePayment'); Not Null	How the customer paid.
sale_status	VARCHAR2(15)	Default 'Completed'; CHECK in ('Completed','Returned','Cancelled'); Not Null	Current sale status.
discount_amount	NUMBER(10,2)	Default 0; CHECK discount_amount >= 0; Not Null	Discount applied at sale-header level.
total_amount	NUMBER(12,2)	CHECK total_amount >= 0; Not Null	Final recorded sale total.

6.11 sale_items

Column	Data Type	Constraints	Description
sale_item_id	NUMBER	Primary Key; identity; Not Null	Unique sale-line identifier.
sale_id	NUMBER	Foreign Key -> sales.sale_id; Not Null	Sale header to which the line belongs.
product_id	NUMBER	Foreign Key -> products.product_id; Not Null	Product sold.
quantity	NUMBER(8)	CHECK quantity > 0; Not Null	Number of units sold.
unit_price	NUMBER(10,2)	CHECK unit_price > 0; Not Null	Selling price used for this transaction.
line_discount	NUMBER(10,2)	Default 0; CHECK line_discount >= 0; Not Null	Discount applied to this sale line.
(sale_id, product_id)	-	Unique pair recommended	A product should normally appear once within a single sale.

6.12 employee_profiles

Column	Data Type	Constraints	Description
employee_id	NUMBER	Primary Key; Foreign Key -> employees.employee_id; Not Null	Shared key identifying the employee. Because it is both PK and FK, each employee can have at most one profile.
date_of_birth	DATE	Not Null	Employee date of birth.
national_id	VARCHAR2(30)	Unique; Not Null	National ID, passport number, or similar personal identifier.
address	VARCHAR2(200)	Not Null	Current residential address of the employee.
emergency_contact_name	VARCHAR2(120)	Not Null	Name of the person to contact in an emergency.
emergency_contact_phone	VARCHAR2(30)	Not Null	Emergency contact telephone number.
profile_status	VARCHAR2(15)	Default 'Active'; CHECK in ('Active','Inactive'); Not Null	Status of the employee profile.

7. Relationship and Cardinality Summary

Parent	Child	Cardinality	Meaning
branches	employees	1 : M	One branch employs many employees.
categories	products	1 : M	One category contains many products.
branches	branch_inventory	1 : M	One branch has many stock records.
products	branch_inventory	1 : M	One product may be stocked in many branches.
suppliers	purchases	1 : M	One supplier can appear in many purchases.
branches	purchases	1 : M	One branch can receive many purchases.
employees	purchases	1 : M	One employee can record many purchases.
purchases	purchase_items	1 : M	One purchase contains many purchase lines.
products	purchase_items	1 : M	One product can appear in many purchase lines.
branches	sales	1 : M	One branch records many sales.
customers	sales	1 : M optional	A registered customer can have many sales; a sale may be walk-in.
employees	sales	1 : M	One employee can handle many sales.
sales	sale_items	1 : M	One sale contains many sale lines.
products	sale_items	1 : M	One product can appear in many sale lines.
employees	employee_profiles	1 : 1	Each employee has one personal profile, and each profile belongs to one employee through a shared primary/foreign key.

8. Data Population Requirements

Students must populate the database with sufficient and meaningful test data so that all required queries return useful results. Data should be realistic and internally consistent.

- Populate every table with sufficient, meaningful data. Use at least 20 records in customers, products, branch_inventory, purchases, purchase_items, sales, and sale_items. Reference tables may contain fewer records where appropriate.
- Use at least 3 branches, at least 5 categories, and at least 12 distinct products across the sample data.

- Create exactly one employee_profiles row for every employee. No employee may have two profiles, and no profile may exist without a matching employee.
- Include both registered-customer sales and walk-in sales (customer_id = NULL).
- Include products that are sold, products that have never been sold, products below reorder level, and products stocked in more than one branch.
- Include purchases with each purchase_status value and sales with each sale_status value.
- The following values must exist somewhere in the data because later query tasks refer to them: branch "Central Branch", category "Electronics", customer "Aylin Demir", supplier "Prime Wholesale", and product "Wireless Mouse" with SKU "ELEC-101".

9. Laboratory Tasks

LAB 2 - Database Creation and Setup

- Create all 12 tables in Oracle Live SQL according to the schema in Section 6.
- Create all primary-key, foreign-key, UNIQUE, NOT NULL, DEFAULT, and CHECK constraints.
- Build the relations among the tables and verify that invalid foreign-key values are rejected.
- Demonstrate at least one ALTER TABLE operation to add or modify a constraint, then restore the required final schema.
- Demonstrate at least one valid INSERT and one invalid INSERT that is rejected by a constraint.
- Keep the final database objects for use in the next laboratory.

LAB 3 - Data Population

- Populate all tables according to Section 8.
- Use INSERT statements and ensure that parent rows are inserted before dependent rows.
- Demonstrate UPDATE and DELETE operations on suitable records while preserving referential integrity.
- Check data quality using SELECT statements, sorting, filtering, and simple aggregate functions.
- Submit the final population script together with evidence that all 12 tables contain valid data.

LABS 4-5 - SQL Queries

Complete the following queries. The expected solution should use appropriate joins, aggregate functions, GROUP BY/HAVING, subqueries, set logic, ordering, and Oracle SQL features where relevant.

1. Retrieve the total number of registered customers in the system.
2. List the product name, SKU, and selling price of all active products in the "Electronics" category with a selling price greater than 100.
3. Show all completed sales made by the customer "Hassan Mahadjir", including sale ID, branch name, sale date, and total amount.
4. Count the number of completed sales recorded by each branch. Display the branch name and count, ordered from highest to lowest.
5. Retrieve all sales made on 20-NOV-2026. Display sale ID, branch, employee, payment method, status, and total amount.
6. List all branch inventory records where quantity_on_hand is below the product reorder_level. Display branch name, product name, current quantity, and reorder level.
7. Retrieve the most expensive active product in the system.
8. For each sale containing more than one distinct product, display the sale ID and a calculated merchandise total using sale_items (quantity * unit_price - line_discount).
9. List the product ID, product name, quantity, unit price, and line discount for all items belonging to sale ID 2.
10. Find the employee who handled the highest number of completed sales. Display employee name, branch, and number of sales.
11. List all products stocked in "Central Branch" together with quantity_on_hand. Include products with zero quantity if such records exist.
12. For purchase ID 5, list each purchased product, quantity, unit cost, and calculated line subtotal.
13. Find all products that have never appeared in sale_items.
14. Find the product with the highest total quantity sold across completed sales.

15. Calculate total completed-sales revenue for each branch. Display branch name and revenue in descending order.
16. Find registered customers who have made more than one completed sale. Display customer name and number of completed sales.
17. Display each supplier and the total value of Received purchases supplied by that supplier. Show only suppliers whose total exceeds 5,000.
18. Use a transaction to increase the quantity_on_hand of one product in two different branches, create a SAVEPOINT, make another inventory update, ROLLBACK to the SAVEPOINT, and then COMMIT. Show the before-and-after values.
19. List every employee together with the employee profile. Display employee name, branch name, job title, date of birth, national ID, emergency contact name, emergency contact phone, and profile status, ordered by employee name.

LAB 6 - Trigger

Create and test a row-level trigger that maintains stock when a sale item is inserted.

- When a new sale_items row is inserted, identify the branch of the related sale and decrease branch_inventory.quantity_on_hand for the corresponding product.
- The trigger must reject the insert if the branch does not have an inventory record for the product or if the available quantity is insufficient.
- The trigger must update branch_inventory.last_stock_update when stock is successfully changed.
- Use RAISE_APPLICATION_ERROR with a clear message for an invalid or insufficient-stock operation.
- Test the trigger with (a) a successful sale-item insert and (b) an insert that requests more stock than is available. Show the inventory values before and after each test.

LAB 7 - Final Assessment and Demonstration

- Present the completed database and all scripts developed during the semester.
- Be prepared to explain the primary keys, foreign keys, composite key, constraints, and relationship cardinalities.
- Be prepared to modify or write SQL statements during the lab assessment.
- Be prepared to explain and demonstrate the trigger created in Lab 6.
- The final database must run without errors in Oracle Live SQL.

10. Submission and Lab Grading

The laboratory component is cumulative. Students are responsible for preserving their scripts and maintaining a working database throughout the semester.

- Lab attendance and participation: 5% of the course grade across the scheduled laboratory sessions.
- Weekly laboratory submissions: 10% of the course grade. Submit the requested SQL scripts and evidence within the deadline announced for each lab.
- Final lab assessment: 5% of the course grade, based on the completed database, demonstration, and in-lab questions.
- Late or incomplete work may receive reduced credit according to the course/lab policy. There is no make-up laboratory session unless the course coordinator announces otherwise.