Faculty of Engineering and Technology
Computer Science Department
Database Systems
COMP333

Course Project

Instructor: Dr. Bassem Sayrafi
Group No.23

Prepared by:

Raghad Jamhour	**1220212**	**sec.1**
Labiba Sharia	**1220228**	**sec.2**

Date: 10/4/2025

Project Idea:

Electronics Store Data Base Management System

Project description

Company Name: Super Star
Owner: Najeh Madi
Contact: 059-926-2251
Super Star is a company specializing in electronic components, computer parts, and related accessories. The goal of this project is to develop a database management system to improve their operations by organizing inventory, managing customer records, and tracking sales efficiently. This system will enhance overall business management, reduce manual workload, and provide better accessibility to important data.

Technology used

The system will be developed using the following technologies:

- Frontend: HTML (structure) & CSS (styling)
- Backend: Python with Flask (server-side logic)
- Database: MySQL (relational database for data storage and management)

Project Requirements
Entities and Attributes:

- Customer
- Employee
- Product
- Order
- Order details
- Supplier
- Purchase order
- Purchase order details
- Payment
- Invoice

Each customer is recorded with a unique identifier, Name (first and last name), phone number, email address, order count, city and shipping address.
Each Employee has a unique identifier, Name (first and last name), role, phone number, email address and hire date. Also, it has a manager id as foreign key. Employees could be hourly working or having a contract. Hourly working employee has hourly wage and number of hours worked while contract employee has contract id, contract start date, contract end date and salary.
Each product has a unique identifier, Name, category, price, stock quantity, stock arrival date and reorder level.
Each order has a unique identifier, Total price, order date, expected received date, and actual received date. Also, it has customer id and employee id as foreign keys.
Each order details have a unique identifier, price, quantity. Also, it have order id and product id as foreign keys.
Each supplier has a unique identifier, Name, email address, and phone number.
Each purchase order has purchase order id, total price, order date, expected received date and actual received date. Also, it has employee id and supplier id as foreign keys.
Each purchase order details have purchase order detail id, price, and quantity. Also, it has purchase order id and product id as foreign keys.

Relationships:

- Customer places Order: A customer can place multiple orders, and each order is placed by exactly one customer.
- Order contains Order_Detail: An order consists of one or more order detail records, and each order detail belongs to exactly one order.
- Order_Detail referes to Product: Each order detail line references exactly one product, while a product can be referenced in many order details.
- Product is referenced in Purchase_Order_Detail: A product can be referenced in multiple purchase order details, and each purchase order detail references exactly one product.
- Purchase_Order contains Purchase_Order_Detail: A purchase order consists of one or more purchase order details, and each purchase order detail belongs to exactly one purchase order.
- Supplier fullfils Purchase_Order: A supplier can fulfill multiple purchase orders, and each purchase order is fulfilled by exactly one supplier.
- Employee prepares Order: Employees process customer orders, where one employee can process multiple orders, and each order is processed by exactly one employee.
- Employee creates Purchase_Order: Employees create purchase orders to suppliers, where one employee can create multiple purchase orders, and each purchase order is created by exactly one employee.
- Employee manages Employee: an employee may manage multiple employees, and each employee (except top management) is managed by exactly one employee.
- An Employee is either an hourly employee or a contract employee (but not both).

Sample Queries

Customer Queries

1. Retrieve the customers in a certain city, displaying their name, ID, and address.
2. Retrieve the information of the most active customer in a specific month based on order count.
3. Retrieve the names of customers who placed more than 5 orders in the last month.
4. Retrieve customer names and contact details for those who made no purchases in the past year.
5. Retrieve the top 10 customers names by total spending, sorted by the highest amount.
6. Retrieve all orders made by a specific customer.
7. Retrieve customers who haven't placed an order in the last 6 months.
8. Retrieve customer names and total amount spent.

Employee Queries

1. Retrieve the names, IDs, and phone numbers of employees with a salary over 4000 and role as Manager.
2. Retrieve the number of orders served by a certain employee.
3. Retrieve all employees along with the total number of orders they processed.
4. Retrieve the names of employees that are hourly paid.

Order Queries

1. Retrieve the orders late beyond the expected delivery date.
2. Retrieve orders with a total amount between 1000 and 1500.
3. Retrieve all orders containing a specific product.
4. Retrieve all orders that contain more than 3 different products.
5. Retrieve the delivery time (in days) per order.

Inventory Queries

1. Retrieve products with stock quantity of at least 12, sorted by price descending.
2. Retrieve products received from supplier with ID = 4.
3. Retrieve products that have been in stock for more than 2 months.
4. Retrieve product details for items with 15 or fewer units in stock.
5. Retrieve total items sold per product category.
6. Retrieve products with stock below the reorder level
7. Retrieve inventory aging report (products in stock for more than 90 days).
8. Retrieve total stock available by category.
9. Retrieve expected restock date for out-of-stock products.

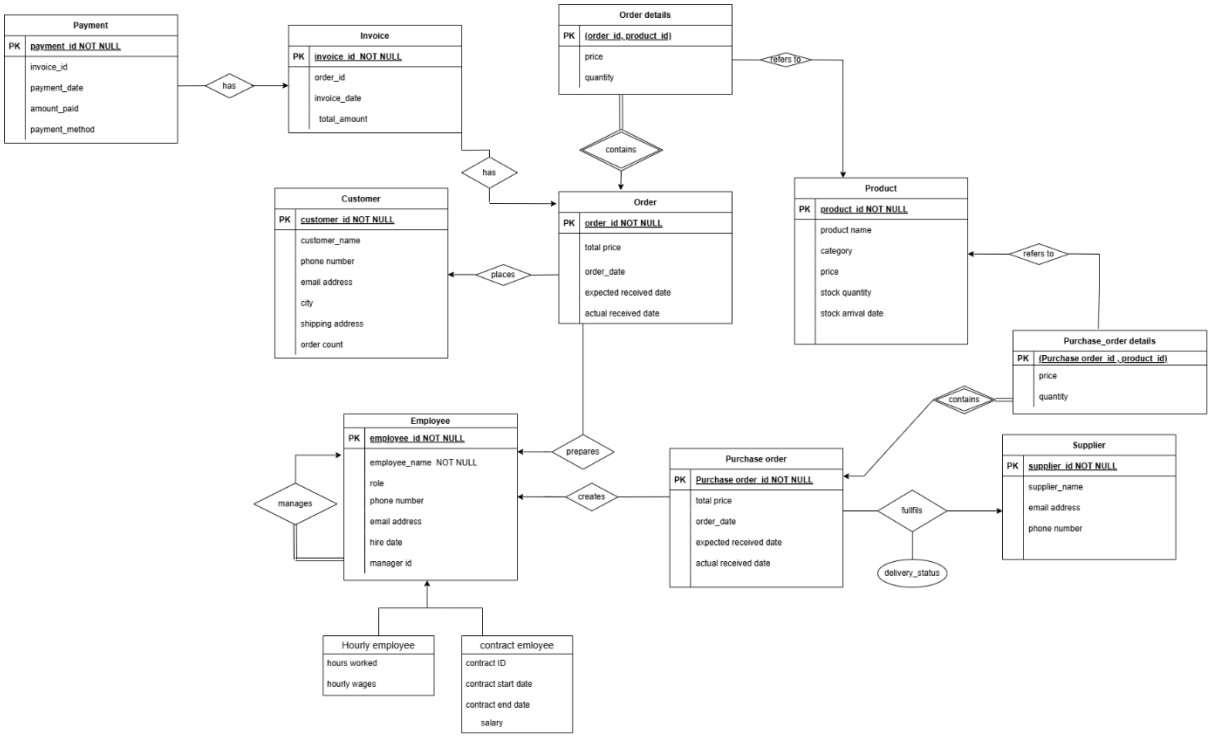
Supplier Queries

1. Retrieve a list of suppliers and the number of products they supply.
2. Retrieve all purchase orders made to a specific supplier.
3. Retrieve suppliers who delivered late orders.
4. Retrieve total spending on purchase orders per supplier.

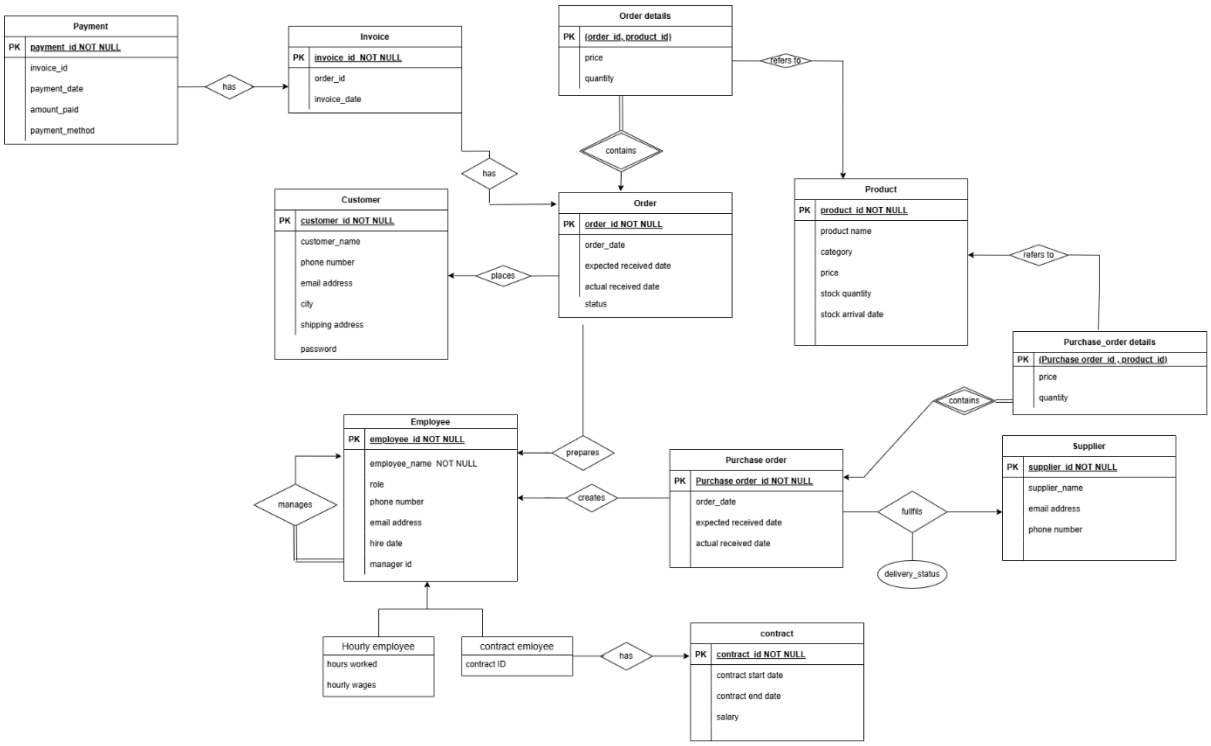
Purchase Order Queries

1. Retrieve all purchase orders and their total amounts.
2. Retrieve products ordered in each purchase order with quantity and unit price.
3. Retrieve purchase orders created by a specific employee.
4. Retrieve delayed purchase orders (where actual delivery is after expected).
5. Retrieve suppliers who have more than 5 purchase orders in the current year.

ER Diagram Before Normalization



ER Diagram After Normalization



1NF (First Normal Form)
All tables have atomic values. Therefore, they satisfy 1NF.

2NF (Second Normal Form)

- Most tables have single-attribute primary keys, so all non-key attributes are fully dependent on the primary key.
- For tables with composite keys (e.g., OrderDetails, PurchaseOrderDetails), all non-key attributes are fully functionally dependent on the entire composite key.

3NF (Third Normal Form)
To satisfy 3NF, for every functional dependency ($X \rightarrow A$), **one** of the following must hold:

1. A is a trivial attribute (i.e., $A \in X$)
2. X is a super key
3. A is part of a candidate key

Most tables meet 3NF because in all functional dependencies, X is the primary key, which is a super key. However, the Contract Employee table violates 3NF due to a transitive dependency.

Functional Dependencies Analysis

- **Customer Table**
  - PK**:** customer_id
  - FD**:** customer_id → customer_name, phone_number, email_address, city, shipping_address, order_count, Customer_password, is_valid
  - All non-key attributes are fully dependent on the primary key.
- **Employee Table**
  - PK**:** employee _id
  - FD**:** employee_id → employee_name, emp_role, phone_number, email_address, hire_date, manager_id, password, is_valid
  - All non-key attributes are fully dependent on the primary key.
- **HourlyEmployee Table**
  - PK**:** employee _id
  - FD**:** employee_id → hours_worked, hourly_wages, is_valid
  - All non-key attributes are fully dependent on the primary key.

  Analysis: This is a specialization table. All attributes are directly dependent on employee_id.
- **Contract Employee Table**
  - PK**:** employee _id
  - FD1: employee_id → contract_id, contract_start_date, contract_end_date, salary, is_valid
  - FD2: contract_id → contract_start_date, contract_end_date, salary

This table violates 3NF due to a transitive dependency:

- employee_id → contract_id
- contract_id → contract_start_date, contract_end_date, salary
- Therefore: employee_id → contract_start_date, contract_end_date, salary (transitive)

Solution: Decomposition into 2 tables: Contract Table & Contract Employee Table

- **Contract Table**
  - PK**:** contract_id
  - FD: contract_id → contract_start_date, contract_end_date, salary, is_valid

- **Contract Employee Table**
  - PK**:** employee _id
  - FD: employee_id → contract_id, is_valid

- **Product Table**
  - PK**:** product _id
  - FD**:** product_id → product_name, category, price, is_valid, stock_quantity, stock_arrival_date
  - All non-key attributes are fully dependent on the primary key.

- **Order Table**
  - PK**:** order _id
  - FD**:** order_id → customer_id, employee_id, order_date, expected_received_date, actual_received_date, is_valid
  - All non-key attributes are fully dependent on the primary key.

- **OrderDetails Table**
  - PK**:** (order_id, product_id) - Composite Key
  - FD**:** (order_id, product_id) → price, quantity, is_valid
  - All non-key attributes are fully dependent on the complete composite key.

- **Supplier Table**
  - PK**:** supplier_id
  - FD**:** supplier_id → supplier_name, email_address, phone_number, is_valid
  - All non-key attributes are fully dependent on the primary key.

- **PurchaseOrder Table**
  - PK**:** purchase_order_id
  - FD**:** purchase_order_id → employee_id, supplier_id, order_date, expected_received_date, actual_received_date, delivery_status, is_valid
  - All non-key attributes are fully dependent on the primary key.

- **PurchaseOrderDetails Table**
  - PK**:** (purchase_order_id, product_id) - Composite Key
  - FD**:** (purchase_order_id, product_id) → price, quantity, is_valid
  - All non-key attributes are fully dependent on the complete composite key.
- **Invoice Table**
  - PK**:** invoice_id

- FD**:** invoice_id → order_id, invoice_date, is_valid
- All non-key attributes are fully dependent on the primary key.

- **Payment Table**
  - PK**:** payment _id
  - FD payment_id → invoice_id, payment_date, amount_paid, payment_method, is_valid
  - All non-key attributes are fully dependent on the primary key.

  Note: Attributes like total amount are not stored directly to avoid redundancy. Instead, they are computed when needed, which follows best normalization practices.

Conversion to relational Tables

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100) NOT NULL,
    phone_number VARCHAR(20),
    email_address VARCHAR(100),
    city VARCHAR(50),
    shipping_address VARCHAR(200) NOT NULL,
    order_count INT DEFAULT 0 CHECK (order_count >= 0),
    Customer_password VARCHAR(100) NOT NULL,
    is_valid BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE Employee (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(100) NOT NULL,
    emp_role VARCHAR(50) CHECK (emp_role IN ('Manager',  'Assistant
Manager', 'Customer Service', 'Procurement Specialist', 'Inventory
Specialist')),
    phone_number VARCHAR(20),
    email_address VARCHAR(100),
    hire_date DATE,
    manager_id INT default -1,
    password VARCHAR(20),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (manager_id) REFERENCES Employee(employee_id)
);


CREATE TABLE HourlyEmployee (
    employee_id INT PRIMARY KEY,
    hours_worked INT NOT NULL CHECK (hours_worked >= 0),
    hourly_wages DECIMAL(10,2) NOT NULL CHECK (hourly_wages >
0),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id)
);

CREATE TABLE Contract (
    contract_id INT PRIMARY KEY,
    contract_start_date DATE NOT NULL,
    contract_end_date DATE NOT NULL,
```

```
    salary DECIMAL(10,2) NOT NULL CHECK (salary > 0),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE
);
CREATE TABLE ContractEmployee (
    employee_id INT PRIMARY KEY,
    contract_id INT NOT NULL,
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id),
    FOREIGN KEY (contract_id) REFERENCES Contract(contract_id)
);
CREATE TABLE Product (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10,2) NOT NULL CHECK (price > 0),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    stock_quantity INT NOT NULL CHECK (stock_quantity >= 0),
    stock_arrival_date DATE
);

CREATE TABLE Product_Archive (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10,2),
    stock_quantity INT,
    stock_arrival_date DATE,
    archived_at DATETIME DEFAULT NOW()
);

CREATE TABLE `Order` (
    order_id INT PRIMARY KEY,
    customer_id INT NOT NULL,
    employee_id INT,
    order_date DATE NOT NULL,
    expected_received_date DATE,
    actual_received_date DATE,
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id)
);

CREATE TABLE OrderDetails (
```

```
    order_id INT,
    product_id INT,
    price DECIMAL(10,2) NOT NULL CHECK (price > 0),
    quantity INT NOT NULL CHECK (quantity > 0),
    PRIMARY KEY (order_id, product_id),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (order_id) REFERENCES `Order`(order_id) on delete
cascade,
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

CREATE TABLE Supplier (
    supplier_id INT PRIMARY KEY,
    supplier_name VARCHAR(100) NOT NULL,
    email_address VARCHAR(100),
    phone_number VARCHAR(20),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE PurchaseOrder (
    purchase_order_id INT PRIMARY KEY,
    employee_id INT NOT NULL,
    supplier_id INT NOT NULL,
    order_date DATE NOT NULL,
    expected_received_date DATE,
    actual_received_date DATE,
    delivery_status VARCHAR(50) DEFAULT 'Pending'
        CHECK (delivery_status IN ('Pending', 'Shipped', 'Received')),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id),
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id)
);

CREATE TABLE PurchaseOrderDetails (
    purchase_order_id INT,
    product_id INT,
    price DECIMAL(10,2) NOT NULL CHECK (price > 0),
    quantity INT NOT NULL CHECK (quantity > 0),
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    PRIMARY KEY (purchase_order_id, product_id),
    FOREIGN KEY (purchase_order_id) REFERENCES
PurchaseOrder(purchase_order_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
```

```
);

CREATE TABLE Invoice (
    invoice_id INT PRIMARY KEY,
    order_id INT NOT NULL,
    invoice_date DATE NOT NULL,
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (order_id) REFERENCES `Order`(order_id)
);

CREATE TABLE Payment (
    payment_id INT PRIMARY KEY,
    invoice_id INT NOT NULL,
    payment_date DATE NOT NULL,
    amount_paid DECIMAL(10,2) NOT NULL CHECK (amount_paid >= 0),
    payment_method VARCHAR(50) NOT NULL,
    is_valid BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (invoice_id) REFERENCES Invoice(invoice_id)
);
```