**Yasmeem Agricultural Company**

**Report for Project**

Prepared by:

**Qusay Mohammad Bdier**

Date: **6/10/2025**

This project involves the development of a comprehensive digital solution for an agricultural tools and equipment store. The system will provide functionalities for inventory management, sales tracking, customer database, supplier management, and an e-commerce platform. A well-structured relational database will be designed to efficiently manage and store critical business data.

## **Key Features:**

Inventory Management: Monitor stock levels, send reorder notifications, and coordinate with suppliers.

Sales & Billing System: Process sales, generate invoices, and record payments.

Customer Management System: Store customer data, track purchases, and facilitate customer support.

Supplier Management: Monitor supplier details, orders, and deliveries.

E-commerce Platform: Online store where customers can browse products, place orders, and make payments.

User Roles & Permissions: Different access levels for administrators, employees, and customers.

Reports & Analytics: Generate sales reports, stock summaries, and financial insights.

1. **Client Information**

Company Name: شركة الياسمين الزراعية

Contact Person: Ahmad Gomaa

Phone Number: 0597141816

Email: Ahmadgomaa@gmail.com

Address: Tulkarum

1. **Technology Stack**

To ensure efficiency, security, and scalability, the following technologies will be used:

Backend: PHP

Frontend: HTML,CSS,JS ,BOOTSTRAP (for an interactive user interface)

Database: MySQL (for structured data storage and relational operations)

Version Control: Git and GitHub for collaboration

## **Queries**

Below are 35 sample MySQL queries.

### **Branch & Staff Queries:**

1. Retrieve details of all branches located in a specific city.
2. Retrieve names, positions, and salaries of staff members at a specific branch, sorted by staff name.
3. Retrieve names of all managers at each branch, sorted by branch number.
4. Retrieve contact details of all employees working in a specific branch.
5. Count the total number of employees in each branch.

### **Product & Inventory Queries:**

1. Retrieve the title, category, and availability of all products at a specified branch, sorted by category.
2. Retrieve the title, category, and availability of all products supplied by a specific supplier at a specified branch, sorted by warehouse entry date.
3. Retrieve the total number of products in each category at a given branch, sorted by supplier.
4. Identify products that are low in stock (below a set threshold) and need reordering.
5. Retrieve product details based on a keyword search (e.g., 'tractor').

### **Sales & Transactions Queries:**

1. Retrieve the total sales made in a specific month.
2. Retrieve details of all sales transactions within a date range.
3. Retrieve the total revenue generated by each branch.
4. Retrieve the most purchased products in the last three months.
5. Retrieve the average sales per customer.

### **Customer Queries:**

1. Retrieve a list of customers who have made at least one purchase.
2. Retrieve customer details along with their purchase history.
3. Identify the top customers based on total spending.
4. Retrieve customers who have not made a purchase in the last six months.
5. Retrieve all orders placed by a specific customer.

### **Supplier Queries:**

1. Retrieve the list of all active suppliers.
2. Retrieve supplier details along with the products they supply.
3. Retrieve the most frequently used suppliers based on order volume.
4. Retrieve pending supplier orders.
5. Retrieve total expenditures on each supplier within the last year.

### **E-commerce Queries:**

1. Retrieve the list of products added to the online store.
2. Retrieve all orders placed through the e-commerce platform.
3. Retrieve payment status of all online transactions.
4. Retrieve the most viewed products in the online store.
5. Retrieve the number of orders completed through the online platform per month.

### **Reports & Analytics Queries:**

1. Generate a monthly stock summary report.
2. Retrieve total revenue per product category.
3. Retrieve sales trends for the past year.
4. Retrieve the best-performing branch based on sales.
5. Retrieve a breakdown of payment methods used by customers.

## **Relations of tables: -**

| **Table** | **Related To** | **Relationship Type** |
| --- | --- | --- |
| Product | Product\_Offers, Order\_Details  Warehouse\_Product | Many-to-One |
| Product | Supplier, Category, Branch\_product | One-to-Many (via PK) |
| Product | Branch | Many-to-Many |
| Order\_Table | Customer, Employee | Many-to-One |
| Order\_Details | Order\_Table, Product | Many-to-Many |
| Cart | Customer | One-to-One |
| Cart\_Items | Cart, Product | Many-to-Many |
| Employee | Account\_Table | One-to-One / Self-Join (Manager) |
| Employee | Branch, Employee | Many-to-One |
| Customer | Account\_Table, Visa | One-to-One |
| Offers | Employee | Many-to-One |
| Product\_Offers | Product, Offers | Many-to-Many |
| Product\_Rating | Product | Many-to-One |

## **Tables:-**

### **Branch Table**

**Purpose:**Stores information about company branches, including location details and contact information.

**Relationships:**Standalone table, can be linked to employees, inventory, or sales for branch-specific data.

**Attributes:**

* branch\_id (primary key)
* name
* address
* city
* street
* phone\_number

### **Branch\_Product Table**

**Purpose:**Tracks stock quantities and availability status of products at different company branches, facilitating branch-level inventory management.

**Relationships:  
Each record links:**

* A product (product\_id)
* A branch (branch\_id)  
  Both foreign keys cascade on delete and update.

**Attributes:**

* branch\_product\_id (primary key)
* product\_id (foreign key)
* branch\_id (foreign key)
* total\_stock\_quantity
* product\_condition

### **Account\_Table**

**Purpose:**  
Stores detailed personal and authentication information for all system users, including customers, employees, or admins, depending on their account\_type.

**Relationships:**  
This is a foundational table that can be referenced by other entities (e.g., login sessions, access control, or extended profile types like customers or employees).

**Attributes:**

* account\_id (primary key)
* first\_name
* last\_name
* email (unique)
* phone\_number
* account\_type
* image
* date\_of\_birth
* password\_hash
* created\_at
* is\_active
* gender
* city
* address

### **Employee Table**

**Purpose:**  
Stores employee-specific data such as job position, salary, hire date, and organizational structure (e.g., manager relationships).

**Relationships:**

* Each employee is linked to an account in Account\_Table via account\_id.
* Assigned to a branch via branch\_id.
* May report to another employee via manager\_id.
* Deleting an account or branch removes the employee record (cascade).
* If a manager is deleted, manager\_id is set to NULL (preserving the employee record).

**Attributes:**

* employee\_id (primary key)
* account\_id (foreign key, unique)
* branch\_id (foreign key)
* manager\_id (foreign key, self-referencing)
* position
* salary
* hire\_date

### **Supplier Table**

**Purpose:**  
Stores contact and identity information of product suppliers for order fulfillment and inventory sourcing.

**Relationships:**  
This is a standalone table that can be linked to a product supply system (e.g., a Product\_Supplier mapping table).

**Attributes:**

* supplier\_id (primary key)
* first\_name
* last\_name
* phone\_number
* email
* address

### **Product Table**

**Purpose:** Stores detailed information about the products available for sale, including pricing, descriptions, and performance metrics like rating and view count.

**Relationships:  
Each product is:**

* Supplied by a Supplier via supplier\_id
* Assigned to a Category via category\_id  
  If a supplier or category is deleted, associated products are also deleted (cascade delete).  
  Updates to referenced IDs are cascaded.

**Attributes:**

* product\_id (primary key)
* supplier\_id (foreign key)
* category\_id (foreign key)
* name
* price
* wholesale\_price
* image
* description\_product
* rate
* views\_count

### **Warehouse Table**

**Purpose:**Stores information about company warehouses used for storing and managing inventory across different locations.

**Relationships:**Currently a standalone table, but it can be linked to inventory, product, or logistics systems.

**Attributes:**

* warehouse\_id (primary key)
* name
* phone\_number
* address
* city

### **Warehouse\_Product Table**

**Purpose:**Tracks the stock quantities of each product stored in different warehouses, enabling inventory management across multiple locations.

**Relationships:**  
Each record links:

* A product (product\_id)
* A warehouse (warehouse\_id)  
  Both foreign keys cascade on delete and update.

**Attributes:**

* product\_id (foreign key, part of composite primary key)
* warehouse\_id (foreign key, part of composite primary key)
* total\_stock\_quantity

Primary Key: Composite key (product\_id, warehouse\_id)

### **Branch Product Inventory Table**

**Purpose**: Stores the quantity of each product **available at each branch**.

**Attributes:**

* branch\_id (primary key, foreign → Branch.branch\_id)
* product\_id (primary key, foreign → Product.product\_id)
* total\_stock\_quantity

### **Warehouse Product Inventory Table**

**Purpose**: Handles the many-to-many relationship between Warehouseand products.

**Attributes:**

* warehouse\_id (primary key, foreign → Warehouse.warehouse\_id)
* product\_id (primary key, foreign → Product.product\_id)
* total\_stock\_quantity

### **Customer Table**

**Purpose**: Stores customer information and purchase history.

**Relationships:**

Customers **place orders** (customer\_id → **Order Table**).

Customers receive **notifications** (customer\_id → **Notifications Table**).

Customers interact with products online (customer\_id → **E-commerce Activity Table**).

**Attributes:**

* customer\_id (primary key)
* registration\_date

1. **Category Table**  
   **Purpose:**  
   Defines the various product categories to organize items for easier browsing and management.

**Relationships:**  
This table is standalone but typically serves as a parent entity for products. Products can be assigned to one category via a foreign key (not shown here but commonly implemented in the Product table).

**Attributes:**

* category\_id (primary key)
* category\_name
* image

1. **Order\_Table**  
   **Purpose:**  
   Stores information about customer orders, including assigned employees (if any), delivery address, total cost, and payment method.

**Relationships:**  
Each order is associated with:

* A customer via customer\_id (foreign key).
* Optionally, an employee via employee\_id (foreign key).  
  Deleting a customer or employee will also remove related orders (cascade delete).  
  Updates to their IDs will propagate to this table (cascade update).

**Attributes:**

* order\_id (primary key)
* customer\_id (foreign key)
* employee\_id (foreign key, optional)
* order\_date
* total\_amount
* city
* street
* apt\_number
* payment\_method
* status\_of\_order

1. **Order\_Details Table**  
   **Purpose:**  
   Represents the details of individual products within a specific customer order, including quantity and the date they were added.

**Relationships:**  
Each record links a product to an order using foreign keys:

* product\_id references the Product table
* order\_id references the Order\_Table  
  If a product or order is deleted, associated entries in this table are also deleted (cascade delete).  
  Updates to product or order IDs will propagate (cascade update).

**Attributes:**

* product\_id (foreign key, part of primary key)
* order\_id (foreign key, part of primary key)
* quantity
* added\_at

**Primary Key:** Composite key (product\_id, order\_id)

### **Offers Table**

**Purpose:** Stores promotional offers created by employees, including discount details and validity periods.

**Relationships:**

Each offer is created by an employee, linked via the employee\_id foreign key.  
If the employee is deleted, their associated offers will also be deleted (cascade delete).  
Updates to the employee ID will reflect in this table (cascade update).

**Attributes:**

* offers\_id (primary key)
* employee\_id (foreign key)
* discount\_percentage
* start\_date
* end\_date
* created\_by
* status\_offer
* title

1. **Product\_Offers Table**  
   **Purpose:** Establishes a many-to-many relationship between products and promotional offers, allowing multiple products to be linked to multiple offers.

**Relationships:**

Each entry connects a product and an offer using foreign keys:

* product\_id references the Product table
* offers\_id references the Offers table  
  Deleting a product or offer will automatically remove associated records in this table (cascade delete).  
  Updates to either key will also propagate (cascade update).

**Attributes:**

* product\_id (foreign key, part of primary key)
* offers\_id (foreign key, part of primary key)

**Primary Key:** Composite key (product\_id, offers\_id)

1. **Visa\_Table**  
   **Purpose:**  
   Stores visa-related information for customers who require international travel or work permissions.

**Relationships:**  
Each visa is linked to a specific customer via the customer\_id foreign key.  
If a customer is deleted, their associated visa records are also deleted (cascade delete).  
Updates to the customer ID will cascade to this table.

**Attributes:**

* visa\_id (primary key)
* visa\_number
* visa\_type
* expiry\_date
* status
* customer\_id (foreign key)

1. **Product\_Rating Table**

**Purpose:** Stores customer ratings for products, allowing analysis of product feedback and satisfaction.

**Relationships:**Each rating is associated with a product via the product\_id foreign key.  
Deleting a product will also delete its associated ratings (cascade delete).

**Attributes:**

* id (primary key)
* product\_id (foreign key)
* rating
* created\_at

1. **Cart Table**

**Purpose:** Represents a shopping cart created by a customer, used to store items before checkout.

**Relationships:**  
Each cart belongs to a specific customer, linked through the customer\_id foreign key.  
If a customer is deleted, their carts are also deleted (cascade delete).  
Customer updates will reflect on associated carts (cascade update).

**Attributes:**

* cart\_id (primary key)
* customer\_id (foreign key)
* created\_at

1. **Cart\_Items Table**

**Purpose:** Stores individual product entries in a customer's cart, including quantity and timestamp.

**Relationships:**  
Each item is linked to a specific cart (cart\_id) and a specific product (product\_id).  
If the cart or product is deleted, the associated cart item is also removed (cascade delete).  
Updates to either referenced key will cascade.

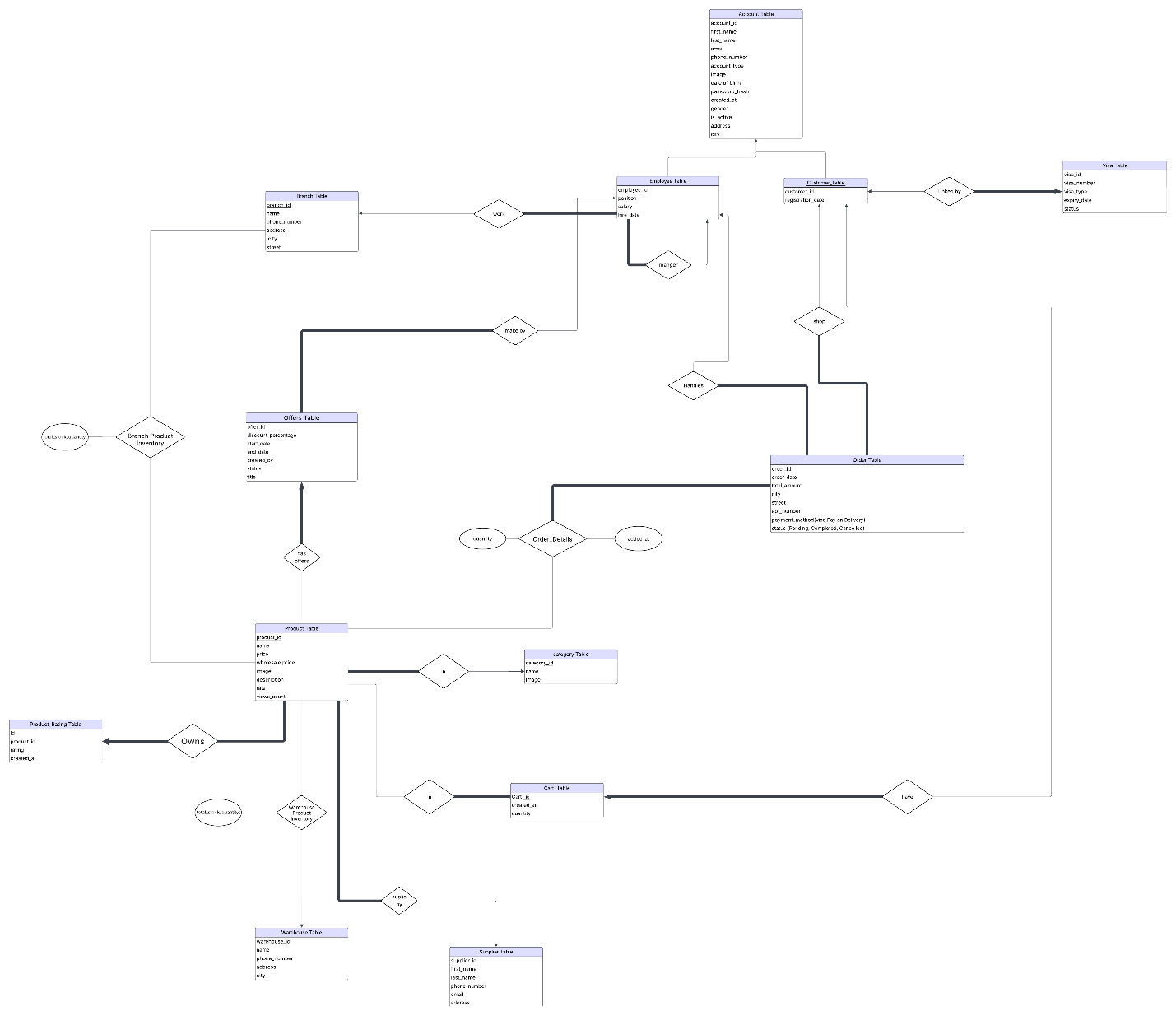
**Attributes:**

* cart\_item\_id (primary key)
* cart\_id (foreign key)
* product\_id (foreign key)
* quantity
* added\_at

## **Normalization:**

1. **Supplier:** The table is in 1NF with atomic fields and no repeating groups. Since the primary key is a single column (supplier\_id), all other fields fully depend on it (2NF). No transitive dependencies exist, so it is in 3NF.
2. **Warehouse:** Atomic fields ensure 1NF compliance. The single-column primary key means all non-key attributes fully depend on it (2NF). No attributes depend on others, so it satisfies 3NF.
3. **Category:** Simple structure with atomic fields (1NF). All attributes depend on the primary key category\_id (2NF), and no transitive dependencies are present (3NF).
4. **Product:** 1NF is satisfied with atomic values. It has foreign keys referencing Supplier and Category ensuring referential integrity. All attributes depend on product\_id fully (2NF), and no transitive dependencies exist (3NF).
5. **Warehouse\_Product:** Composite primary key (product\_id, warehouse\_id) fully determines total\_stock\_quantity (2NF). No non-key attribute depends on another non-key attribute, so 3NF is met.
6. **Offers:** All fields depend on the primary key offers\_id (1NF and 2NF). No transitive dependencies exist; references to employees are by foreign key only (3NF).
7. **Product\_Offers:** Composite primary key ensures full dependency. No other fields exist to violate normalization rules.
8. **Order\_Table:** Atomic fields (1NF), single primary key order\_id with all fields dependent on it (2NF). No transitive dependencies are present (3NF).
9. **Order\_Details:** Composite key (product\_id, order\_id) fully determines the quantity and added date (2NF). No transitive dependencies (3NF).
10. **Branch:** Atomic attributes (1NF). All depends on branch\_id (2NF). No transitive dependencies (3NF).
11. **Branch\_Product:** Composite key with full dependency on stock quantity (2NF). No transitive dependencies (3NF).
12. **Visa\_Table:** Atomic fields (1NF). Primary key fully determines all attributes (2NF). No attribute depends on other non-key attributes, so 3NF.
13. **Account\_Table:** Atomic fields (1NF). Primary key account\_id fully determines attributes (2NF). No transitive dependencies (3NF).
14. **Employee:** Atomic fields (1NF). All fields depend on employee\_id (2NF). Foreign keys ensure relations without transitive dependencies (3NF).
15. **Customer:** Atomic fields (1NF). All attributes depend on customer\_id (2NF). No transitive dependencies (3NF).
16. **Product\_Rating:** Atomic (1NF). Attributes fully depend on id (2NF). No transitive dependencies (3NF).
17. **Cart:** Atomic (1NF). Attributes fully depend on cart\_id (2NF). No transitive dependencies (3NF).
18. **Cart\_Items:** Atomic (1NF). Attributes fully depend on cart\_item\_id (2NF). No transitive dependencies (3NF).

## **ERD Diagram**



\*\*\*Link to visit the site

<https://lucid.app/lucidchart/cb4a1765-4334-47bf-acbf-db29618bb8d3/edit?viewport_loc=2500%2C-1912%2C11744%2C4831%2C0_0&invitationId=inv_852733e5-6686-48bf-a16d-368643d62ffd>