A

Project Report on Analysis, Design and Implementation Of Database for Nursery Store

Chapters

Chapter 1: Introduction

1.1 Introduction

In today's digital age, effective management of data is vital for businesses across various industries. The nursery store industry, in particular, relies heavily on efficient organization and access to data to meet the demands of customers and streamline operations. Recognizing the importance of a robust information system, we present a comprehensive database project specifically tailored for a nursery store.

This project aims to design and implement a database system that caters to the unique requirements of a nursery store, providing an efficient and reliable platform for managing various aspects of the business. By leveraging modern database technologies, we intend to enhance the store's ability to handle various operations.

1.2 Problem Statement

The nursery store faces challenges in manual inventory management, inefficient sales tracking, limited customer relationship management, ineffective supplier management, and the absence of a robust reporting and analytics system. To address these issues, a comprehensive nursery store database system is required. The system aims to streamline operations by centralizing information management, improving inventory control, sales tracking, customer relationship management, supplier management, and reporting and analytics capabilities.

By implementing this database project, the nursery store can overcome these challenges, optimize stock levels, generate accurate sales reports, enhance customer satisfaction, streamline supplier interactions, and make informed decisions based on real-time data. The project seeks to provide an efficient solution that empowers the nursery store to achieve operational excellence, improve customer experiences, and drive growth in a competitive market.

1.3 Objectives

1. Data Organization: The nursery dataset aims to organize and maintain a comprehensive collection of information related to plants, customers, inventory, and sales in a structured and easily accessible manner.
2. Efficient Inventory Management: The dataset seeks to facilitate efficient management of plant inventory by accurately tracking the available stock, plant varieties, growth stages, and supplier information. This enables the nursery to optimize its purchasing, pricing, and restocking processes.
3. Customer Relationship Management: The dataset aims to enhance customer relationship management by storing and analyzing customer information, preferences, purchase history, and communication records. This allows the nursery to provide personalized services, track customer interactions, and foster long-term relationships.

4. Sales Tracking and Analysis: The dataset aims to enable accurate tracking and analysis of sales data, including the quantity and value of plants sold, pricing trends, popular plant varieties, and seasonal demand patterns. This information helps the nursery make informed decisions about pricing, promotions, and sales strategies.
5. Reporting and Analytics: The dataset aims to generate insightful reports and analytics, providing the nursery with valuable information on key performance indicators, sales trends, customer behavior, and inventory metrics. These insights enable data-driven decision-making and strategic planning.
6. Scalability and Adaptability: The dataset aims to be designed to accommodate future growth and adapt to evolving business needs. It is scalable, allowing the addition of new plant varieties, customers, and inventory items. It will also be flexible, enabling customization and integration with other systems or technologies as required.

Chapter 2: Analysis and Design

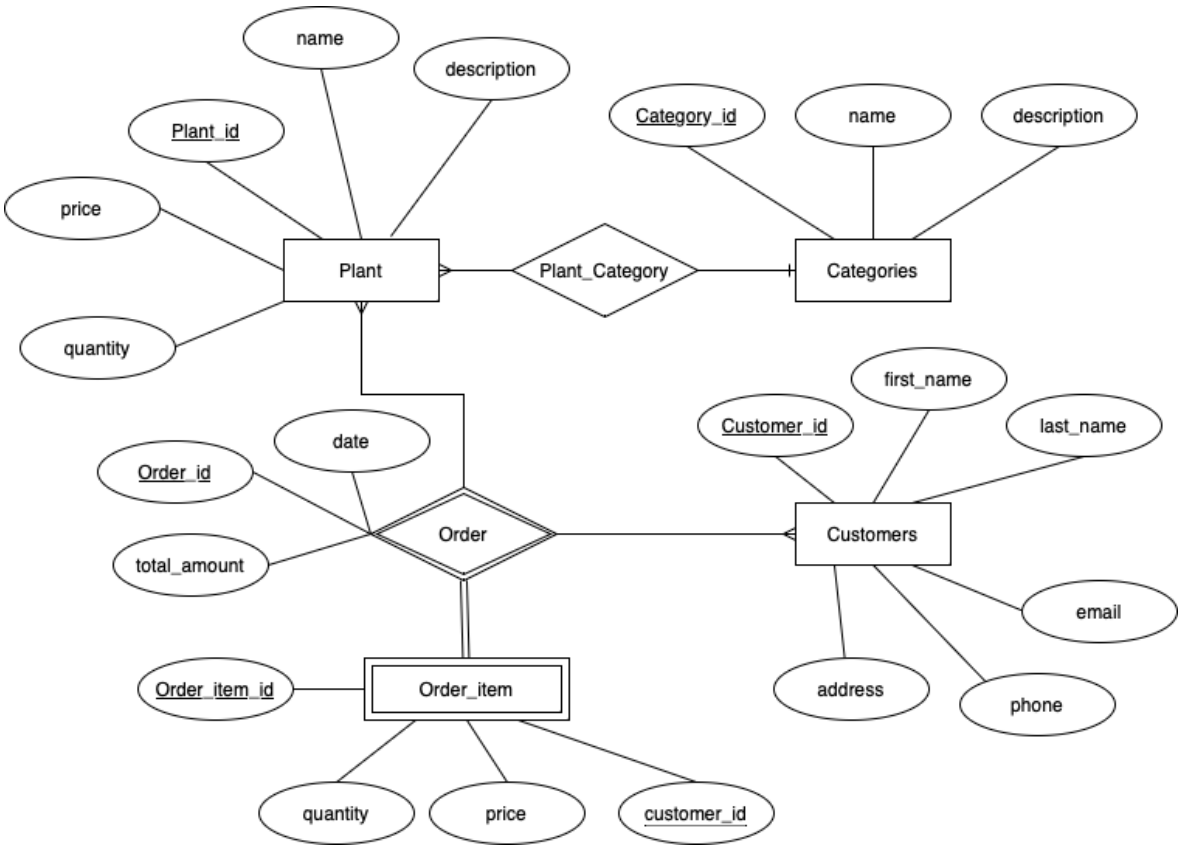
2.1. Analysis

The analysis of the Nursery Store database design involves evaluating its key components, structure, and functionality. Here is an overview of the analysis:

Database Components:

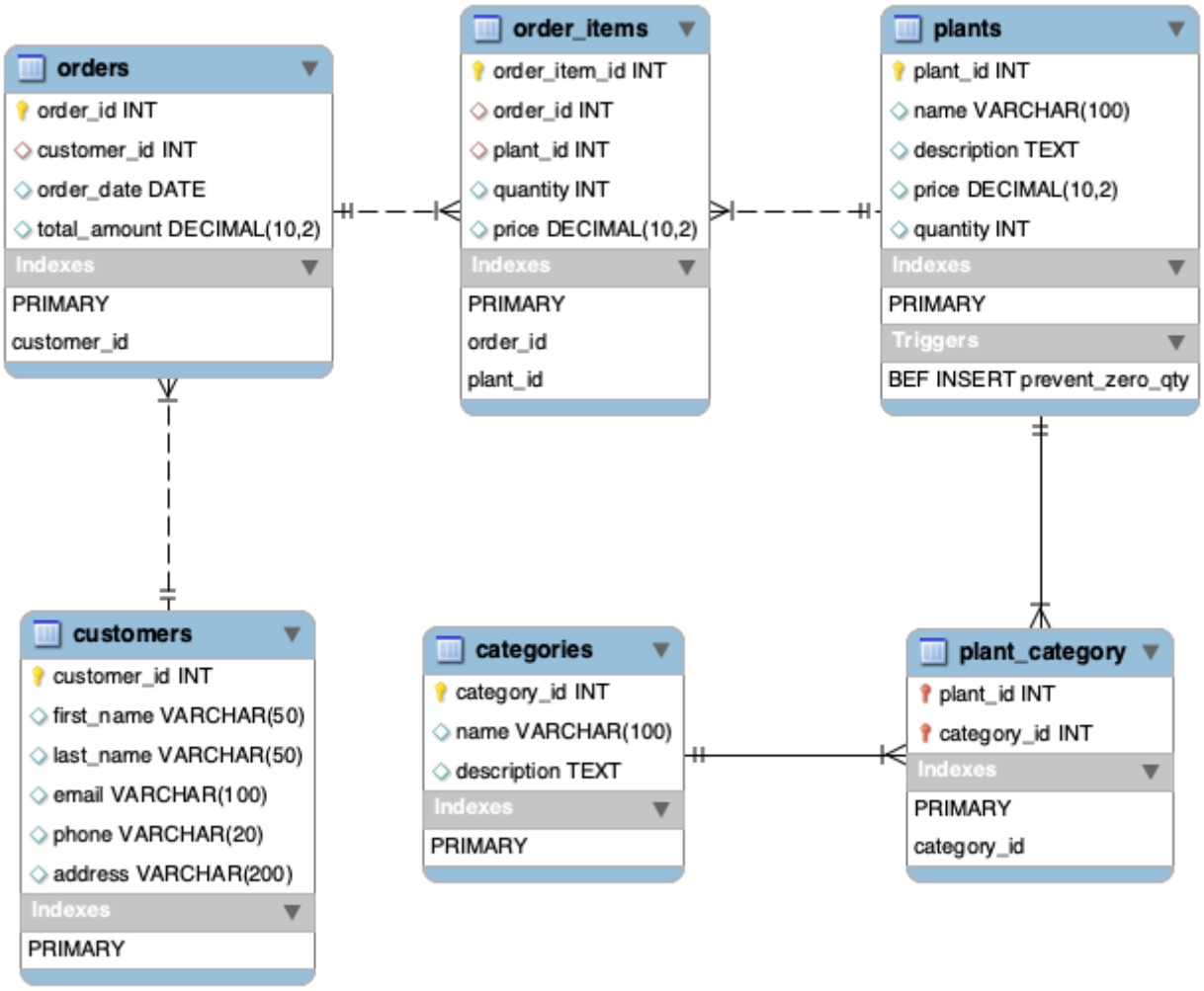
- Tables: The database design should include tables to store essential data entities such as inventory, sales, customers, suppliers, and reports.
- Relationships: The relationships between tables should be properly defined, such as one-to-many relationships between sales and customers or suppliers, and foreign key constraints to ensure data integrity.
- Primary Keys: Each table should have a primary key to uniquely identify records and facilitate efficient data retrieval.

2.1.1. ER Modelling (followed by description)



2.2. Design

2.2.1. Database Design / Schema Diagram (followed by description)



Chapter 3: Implementation

3.1. DDL Statements

```
CREATE DATABASE nursery_store;
USE nursery_store;
```

Create Customer Table

```
CREATE TABLE customers ( customer_id INT PRIMARY KEY AUTO_INCREMENT, first_name
VARCHAR(50), last_name VARCHAR(50), email VARCHAR(100), phone VARCHAR(20),
address VARCHAR(200) );
```

Create Plant Table

```
CREATE TABLE plants ( plant_id INT PRIMARY KEY AUTO_INCREMENT, name
VARCHAR(100), description TEXT, price DECIMAL(10,2), quantity INT );
```

Create Order Table

```
CREATE TABLE orders ( order_id INT PRIMARY KEY AUTO_INCREMENT, customer_id INT,
order_date DATE, total_amount DECIMAL(10,2), FOREIGN KEY (customer_id)
REFERENCES customers(customer_id) );
```

Create Order Item table

```
CREATE TABLE order_items ( order_item_id INT PRIMARY KEY AUTO_INCREMENT,
order_id INT, plant_id INT, quantity INT, price DECIMAL(10,2), FOREIGN KEY
(order_id) REFERENCES orders(order_id), FOREIGN KEY (plant_id) REFERENCES
plants(plant_id) );
```

Create Categories Table

```
CREATE TABLE categories ( category_id INT PRIMARY KEY AUTO_INCREMENT, name
VARCHAR(100), description TEXT );
```

Create Plant Category Table

```
CREATE TABLE plant_category ( plant_id INT, category_id INT, PRIMARY KEY
(plant_id, category_id), FOREIGN KEY (plant_id) REFERENCES plants(plant_id),
FOREIGN KEY (category_id) REFERENCES categories(category_id) );
```

3.2. DML Statements

Insert into Customer Table

```
INSERT INTO customers (first_name, last_name, email, phone, address) VALUES
('John', 'Doe', 'johndoe@example.com', '9843342697', '123 Main Street'),
('Jane', 'Smith', 'janesmith@example.com', '9841778832', '456 Elm Avenue'),
('Michael', 'Johnson', 'michaeljohnson@example.com', '9861298978', '789 Oak
Lane'),
('Emily', 'Brown', 'emilybrown@example.com', '9841667434', '987 Pine Road'),
('David', 'Miller', 'davidmiller@example.com', '9841667674', '654 Cedar
Court'),
('Sarah', 'Wilson', 'sarahwilson@example.com', '98416674557', '321 Birch
Lane'),
('Robert', 'Anderson', 'robertanderson@example.com', '9841567321', '456 Maple
Street'),
('Jessica', 'Thomas', 'jessicathomas@example.com', '9854453467', '78 Sink
Avenue'),
('Christopher', 'Jackson', 'christopherjackson@example.com', '9876846321', '987
Pine Lane'),
('Jennifer', 'White', 'jenniferwhite@example.com', '984672235', '654 Elm
Court');
```

Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.)

```
select * from customers;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	customer_id	first_name	last_name	email	phone	address
Edit Copy Delete	1	John	Doe	johndoe@example.com	9843342697	123 Main Street
Edit Copy Delete	2	Jane	Smith	janesmith@example.com	9841778832	456 Elm Avenue
Edit Copy Delete	3	Michael	Johnson	michaeljohnson@example.com	9861298978	789 Oak Lane
Edit Copy Delete	4	Emily	Brown	emilybrown@example.com	9841667434	987 Pine Road
Edit Copy Delete	5	David	Miller	davidmiller@example.com	9841667674	654 Cedar Court
Edit Copy Delete	6	Sarah	Wilson	sarahwilson@example.com	98416674557	321 Birch Lane
Edit Copy Delete	7	Robert	Anderson	robertanderson@example.com	9841567321	456 Maple Street
Edit Copy Delete	8	Jessica	Thomas	jessicathomas@example.com	9854453467	78 Sink Avenue
Edit Copy Delete	9	Christopher	Jackson	christopherjackson@example.com	9876846321	987 Pine Lane
Edit Copy Delete	10	Jennifer	White	jenniferwhite@example.com	984672235	654 Elm Court

Insert into Plant table

```
INSERT INTO plants (name, description, price, quantity) VALUES
('Rose', 'Beautiful flowering plant', 19.99, 10),
('Lily', 'Elegant and fragrant flower', 12.99, 5),
('Fern', 'Lush green foliage plant', 9.99, 20),
('Cactus', 'Low-maintenance desert plant', 7.99, 15),
('Orchid', 'Exotic and delicate flower', 24.99, 8),
('Bonsai', 'Artfully pruned miniature tree', 39.99, 3),
('Succulent', 'Drought-tolerant and cute', 6.99, 12),
('Palm', 'Tropical indoor palm tree', 29.99, 6),
('Snake Plant', 'Air-purifying and resilient', 14.99, 9),
('Pothos', 'Popular trailing houseplant', 8.99, 18);
```

Showing rows 0 - 9 (10 total, Query took 0.0003 seconds.)

```
select * from plants;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	plant_id	name	description	price	quantity
Edit Copy Delete	1	Rose	Beautiful flowering plant	19.99	10
Edit Copy Delete	2	Lily	Elegant and fragrant flower	12.99	5
Edit Copy Delete	3	Fern	Lush green foliage plant	9.99	20
Edit Copy Delete	4	Cactus	Low-maintenance desert plant	7.99	15
Edit Copy Delete	5	Orchid	Exotic and delicate flower	24.99	8
Edit Copy Delete	6	Bonsai	Artfully pruned miniature tree	39.99	3
Edit Copy Delete	7	Succulent	Drought-tolerant and cute	6.99	12
Edit Copy Delete	8	Palm	Tropical indoor palm tree	29.99	6
Edit Copy Delete	9	Snake Plant	Air-purifying and resilient	14.99	9
Edit Copy Delete	10	Pothos	Popular trailing houseplant	8.99	18

Insert into Order Table

```
INSERT INTO orders (customer_id, order_date, total_amount) VALUES
(1, '2023-06-18', 75.99),
(2, '2023-06-17', 42.50),
(3, '2023-06-19', 115.75),
(1, '2023-06-19', 32.99),
(4, '2023-06-18', 50.00),
(2, '2023-06-19', 89.95),
(5, '2023-06-17', 27.50),
(3, '2023-06-18', 64.75),
(4, '2023-06-19', 75.00),
(1, '2023-06-17', 112.50);
```

Showing rows 0 - 9 (10 total, Query took 0.0007 seconds.)

```
SELECT*from orders;
```

[Edit inline] [Edit] [Create PHP code]

Show all | Number of rows: 25 Filter rows: Search this table Sort by

Extra options

	order_id	customer_id	order_date	total_amount
Edit Copy Delete	1	1	2023-06-18	75.99
Edit Copy Delete	2	2	2023-06-17	42.50
Edit Copy Delete	3	3	2023-06-19	115.75
Edit Copy Delete	4	1	2023-06-19	32.99
Edit Copy Delete	5	4	2023-06-18	50.00
Edit Copy Delete	6	2	2023-06-19	89.95
Edit Copy Delete	7	5	2023-06-17	27.50
Edit Copy Delete	8	3	2023-06-18	64.75
Edit Copy Delete	9	4	2023-06-19	75.00
Edit Copy Delete	10	1	2023-06-17	112.50

Insert into Order_items table

```
INSERT INTO order_items (order_id, plant_id, quantity, price) VALUES
(1, 2, 3, 38.97),
(1, 5, 1, 24.99),
(2, 3, 2, 19.98),
(3, 1, 4, 79.96),
(4, 4, 2, 15.98),
(5, 2, 1, 12.99),
(5, 6, 3, 119.97),
(6, 7, 2, 13.98),
(7, 3, 1, 9.99),
(8, 1, 3, 59.97);
```

Showing rows 0 - 9 (10 total, Query took 0.0003 seconds.)

```
select * from order_items;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	order_item_id	order_id	plant_id	quantity	price
Edit Copy Delete	1	1	2	3	38.97
Edit Copy Delete	2	1	5	1	24.99
Edit Copy Delete	3	2	3	2	19.98
Edit Copy Delete	4	3	1	4	79.96
Edit Copy Delete	5	4	4	2	15.98
Edit Copy Delete	6	5	2	1	12.99
Edit Copy Delete	7	5	6	3	119.97
Edit Copy Delete	8	6	7	2	13.98
Edit Copy Delete	9	7	3	1	9.99
Edit Copy Delete	10	8	1	3	59.97

Insert into categories

```
INSERT INTO categories (name, description) VALUES
('Flowers', 'Various types of beautiful flowering plants'),
('Foliage', 'Plants with lush green foliage'),
('Cacti', 'Low-maintenance desert plants'),
('Orchids', 'Exotic and delicate flowering plants'),
('Bonsai', 'Artfully pruned miniature trees'),
('Succulents', 'Drought-tolerant and unique plants'),
('Palms', 'Tropical indoor palm trees'),
('Air Plants', 'Epiphytic plants that absorb nutrients from the air'),
('Herbs', 'Plants used for culinary or medicinal purposes'),
('Ferns', 'Plants with delicate and feathery fronds');
```

Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.)

```
select * from categories;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	category_id	name	description
Edit Copy Delete	1	Flowers	Various types of beautiful flowering plants
Edit Copy Delete	2	Foliage	Plants with lush green foliage
Edit Copy Delete	3	Cacti	Low-maintenance desert plants
Edit Copy Delete	4	Orchids	Exotic and delicate flowering plants
Edit Copy Delete	5	Bonsai	Artfully pruned miniature trees
Edit Copy Delete	6	Succulents	Drought-tolerant and unique plants
Edit Copy Delete	7	Palms	Tropical indoor palm trees
Edit Copy Delete	8	Air Plants	Epiphytic plants that absorb nutrients from the ai...
Edit Copy Delete	9	Herbs	Plants used for culinary or medicinal purposes
Edit Copy Delete	10	Ferns	Plants with delicate and feathery fronds

Insert into Plant Category

```
INSERT INTO plant_category (plant_id, category_id) VALUES (1, 1), (10, 9), (2, 1), (3, 10), (4, 3), (5, 1), (6, 5), (7, 6), (8, 7), (9, 9);
```

Showing rows 0 - 9 (10 total, Query took 0.0002 seconds.)

```
select *from plant_category;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	plant_id	category_id
Edit Copy Delete	1	1
Edit Copy Delete	2	1
Edit Copy Delete	3	10
Edit Copy Delete	4	3
Edit Copy Delete	5	1
Edit Copy Delete	6	5
Edit Copy Delete	7	6
Edit Copy Delete	8	7
Edit Copy Delete	9	9
Edit Copy Delete	10	9

Select the Plants with their category name and total quantity

Showing rows 0 - 9 (10 total, Query took 0.0005 seconds.)

```
SELECT p.name AS plant_name, c.name AS category_name, p.quantity FROM plants p JOIN plant_category pc ON p.plant_id = pc.plant_id JOIN categories c ON pc.category_id = c.category_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

plant_name	category_name	quantity
Rose	Flowers	10
Lily	Flowers	5
Fern	Ferns	20
Cactus	Cacti	15
Orchid	Flowers	8
Bonsai	Bonsai	3
Succulent	Succulents	12
Palm	Palms	6
Snake Plant	Herbs	9
Pothos	Herbs	18

Select total order amount from the order table

```
select sum(total_amount) as Total from orders;
```

Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)

```
select sum(total_amount) as Total from orders;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Total
686.93

Show all | Number of rows: 25 Filter rows: Search this table

Select Total Plants in Store

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

```
SELECT SUM(quantity) AS total_plants FROM plants;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

total_plants
106

3.3. Stored Procedure

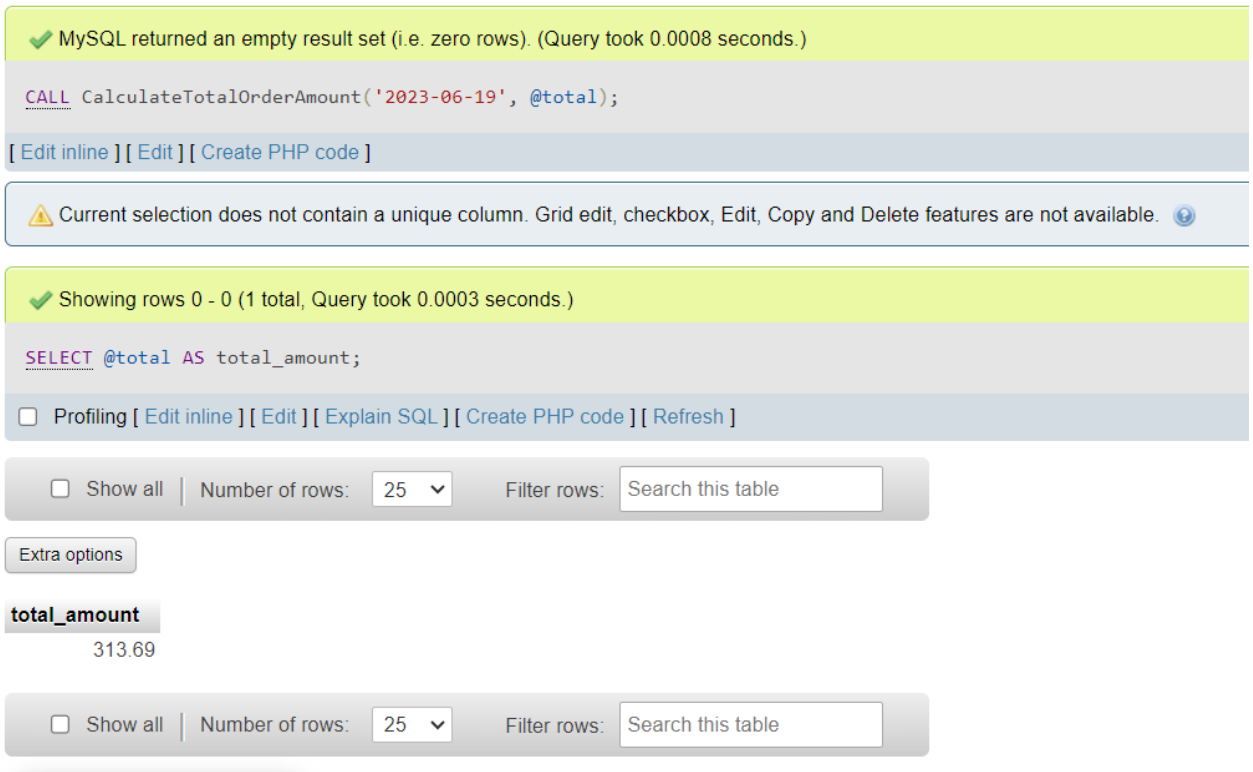
Create store procedure to calculate the total sales of the certain day

```
DELIMITER //

CREATE PROCEDURE CalculateTotalOrderAmount(IN orderDate DATE, OUT totalAmount
DECIMAL(10,2))
BEGIN
   SELECT SUM(total_amount) INTO totalAmount
   FROM orders
   WHERE order_date >= orderDate;
END //

DELIMITER ;
```

```
CREATE PROCEDURE CalculateTotalOrderAmount(IN orderDate DATE, OUT totalAmount
DECIMAL(10,2)) BEGIN SELECT SUM(total_amount) INTO totalAmount FROM orders
WHERE order_date >= orderDate; END;
```

Call the Stored Procedure

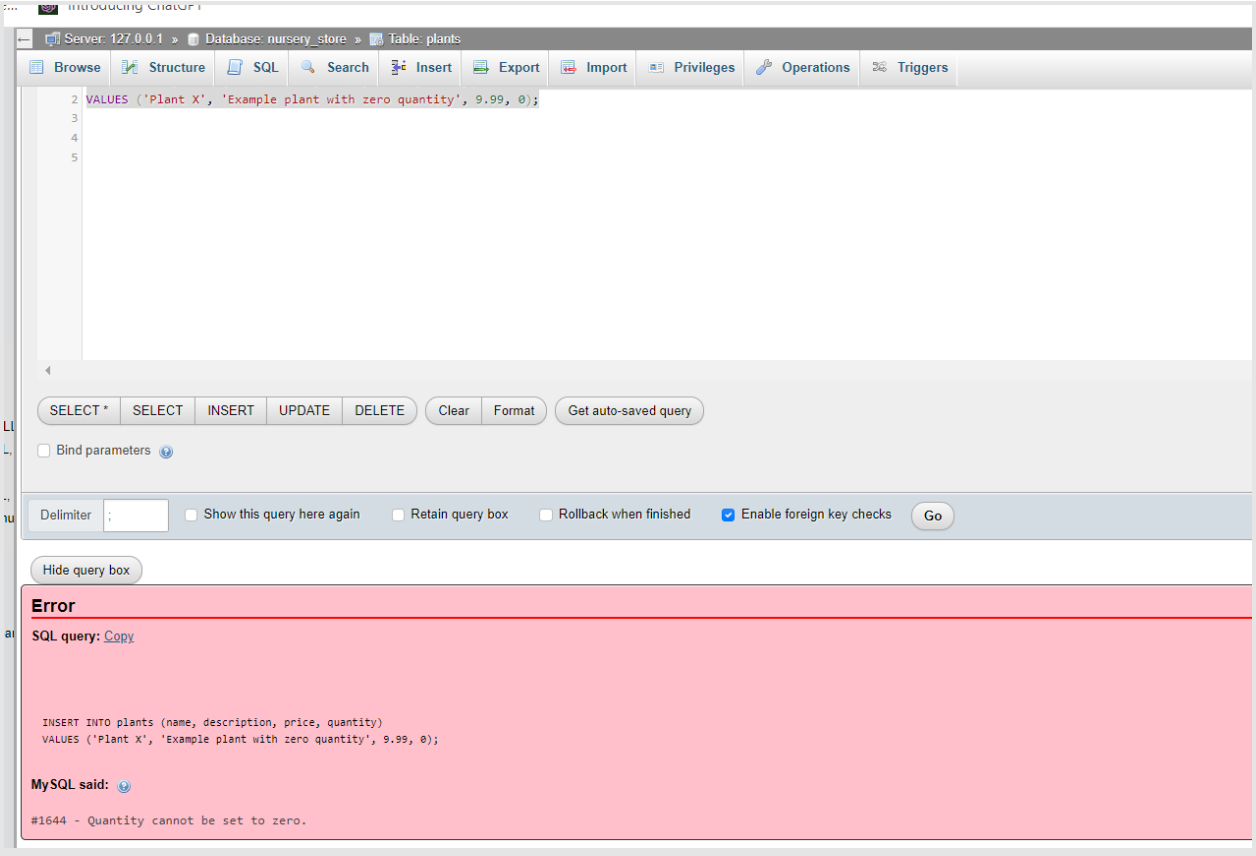
```
CALL CalculateTotalOrderAmount('2023-06-01', @total);
SELECT @total AS total_amount;
```



3.4. Triggers

Quantity cannot be zero while inserting in the plant table

```
DELIMITER //
CREATE TRIGGER prevent_zero_qty
BEFORE INSERT ON plants
FOR EACH ROW
BEGIN
    IF NEW.quantity = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Quantity cannot be set to zero.';
    END IF;
END //
DELIMITER ;
```

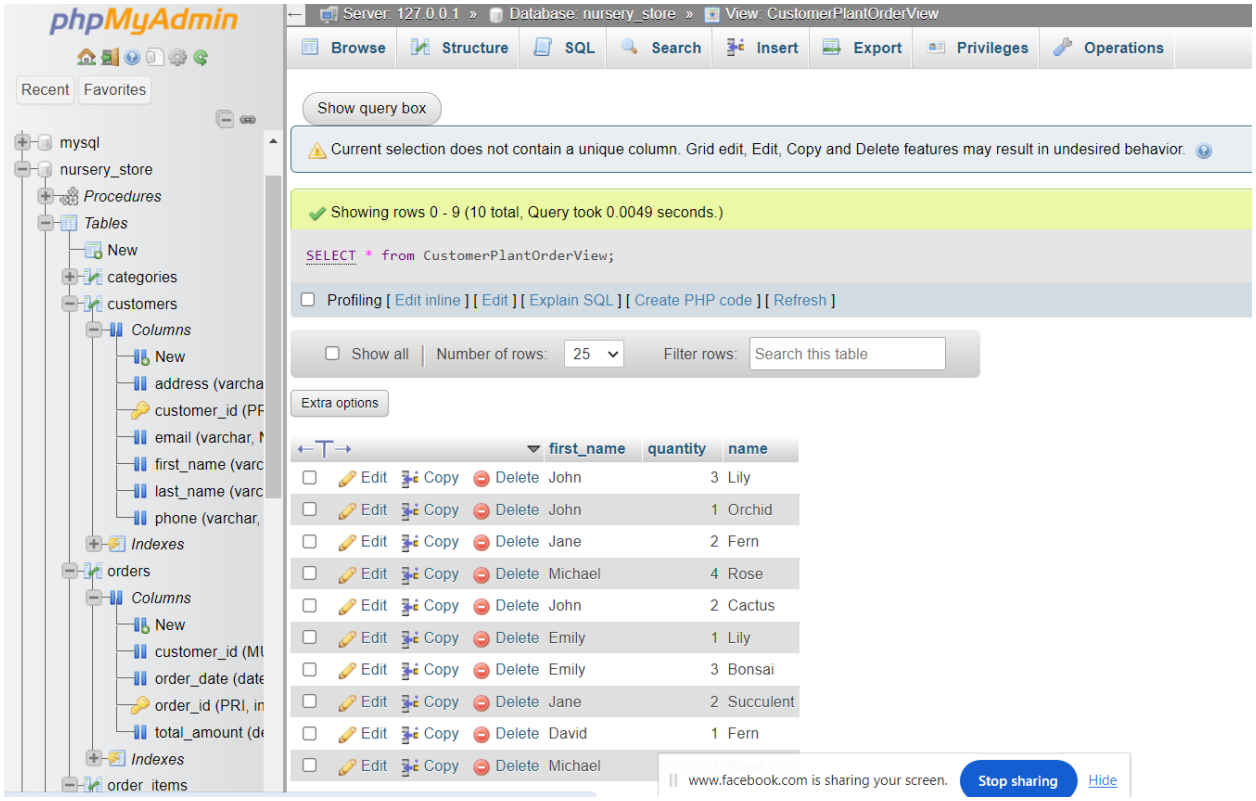


3.5. Views

```
CREATE VIEW CustomerPlantOrderView AS
Select c.first_name,oi.quantity, p.name FROM Customers AS c
JOIN Orders AS o ON o.customer_id = c.customer_id
JOIN order_items as oi ON o.order_id = oi.order_id
JOIN plants as p ON p.plant_id = oi.plant_id;
```

#getting data from view

SELECT * from CustomerPlantOrderView;



This creates a view for customers who bought plants with plant names and quantities.

Chapter 4: Conclusion

We have successfully implemented Nursery Management using nursery data store.