FINAL PROJECT REPORT
DATABASE PRACTICUM

DATABASE INVENTORY MANAGEMENT SYSTEM

Written By :

Al Azhar Rizqi Rifa'i Firdaus

Ichsan Ali Darmawan

Sri Kresna Maha Dewa

Steven Christian Susanto

STUDY PROGRAM D-IV INFORMATICS ENGINEERING

INFORMATION TECHNOLOGY DEPARTMENT

STATE POLYTECHNIC OF MALANG

Soekarno Hatta Street No.9, Jatimulyo, Lowokwaru District, Malang City, East Java

65141

TABLE OF CONTENT

I. INTRODUCTION

An inventory management system is an essential tool for any organization looking to keep track of its inventory-related information effectively. It is a software application that is designed to help organizations manage product, supplier, warehouse, and other inventory-related data efficiently.

In addition to the features mentioned, modern inventory management systems offer a wide range of capabilities that can help organizations streamline their inventory management processes further. Some of these features include:

1. Barcode scanning technology: This allows organizations to track inventory levels in real-time, which can help them make better decisions about inventory management.
2. Automated replenishment: This feature helps organizations to automatically reorder products when inventory levels fall below a certain threshold.
3. Sales forecasting: By analyzing historical sales data, inventory management systems can help organizations predict future demand for products and adjust inventory levels accordingly.
4. Reporting and analytics: Inventory management systems can generate detailed reports and analytics that can help organizations gain insights into their inventory management processes.
5. Integration with other systems: Modern inventory management systems can integrate with other software applications, such as accounting software and supply chain management software, to streamline business operations.

In addition to the benefits mentioned earlier, an inventory management system can also help organizations reduce waste and prevent stockouts. By having accurate and up-to-date information on inventory levels, organizations can avoid overstocking or understocking of products, which can lead to waste or stockouts. This can help organizations save money and reduce their environmental impact by minimizing waste.

Moreover, an inventory management system can help organizations improve customer satisfaction by ensuring that products are available when customers need them. By having real-time information on inventory levels, organizations can quickly respond to customer demands and avoid stockouts, which can damage customer relationships.

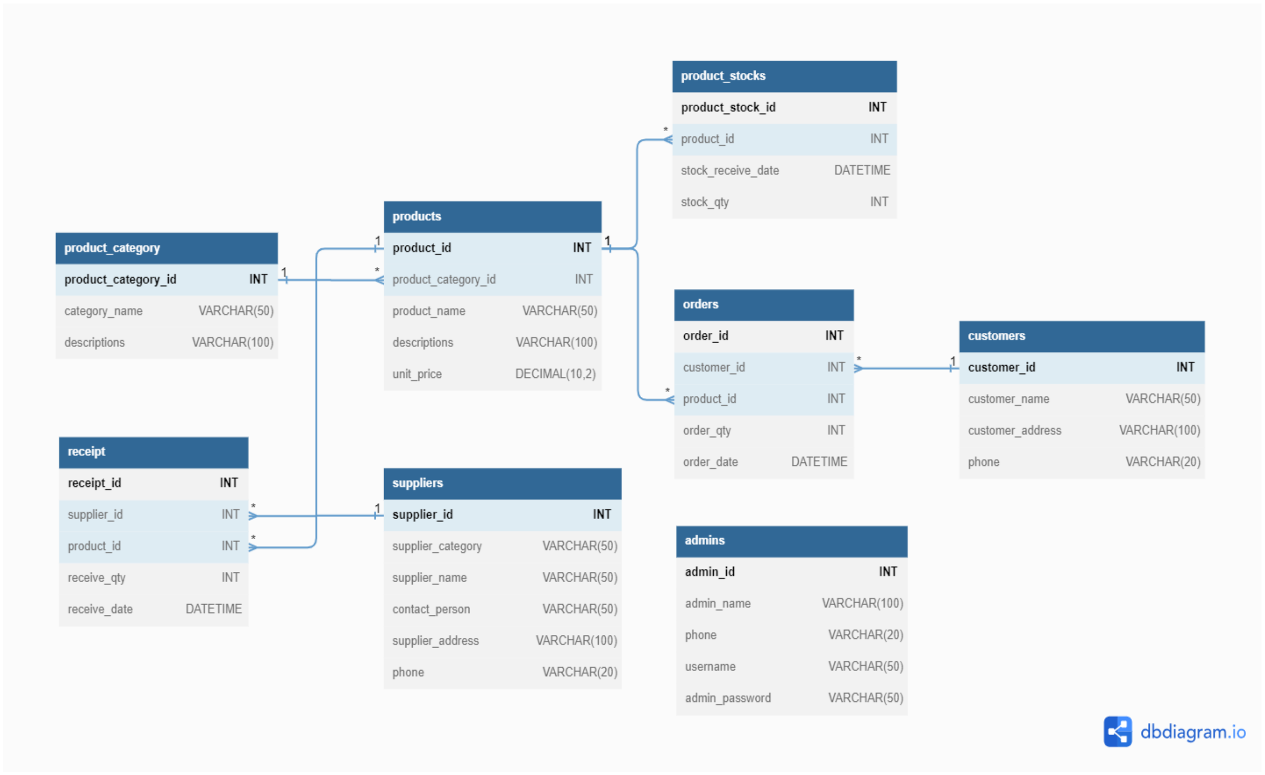
Finally, an inventory management system can help organizations comply with regulatory requirements and industry standards. By tracking information on products, suppliers, and warehouses, organizations can ensure that they are meeting regulatory requirements and industry standards for inventory management.

In conclusion, an inventory management system is an essential tool for any organization looking to manage its inventory-related information effectively and efficiently. With a wide range of features and capabilities, modern inventory management systems can help organizations save time, reduce costs, improve efficiency, and comply with regulatory requirements and industry standards.

In this opportunity our team managed to create a database program that was implemented inside the Inventory Management. Explanation of our program will be explained later the next page.

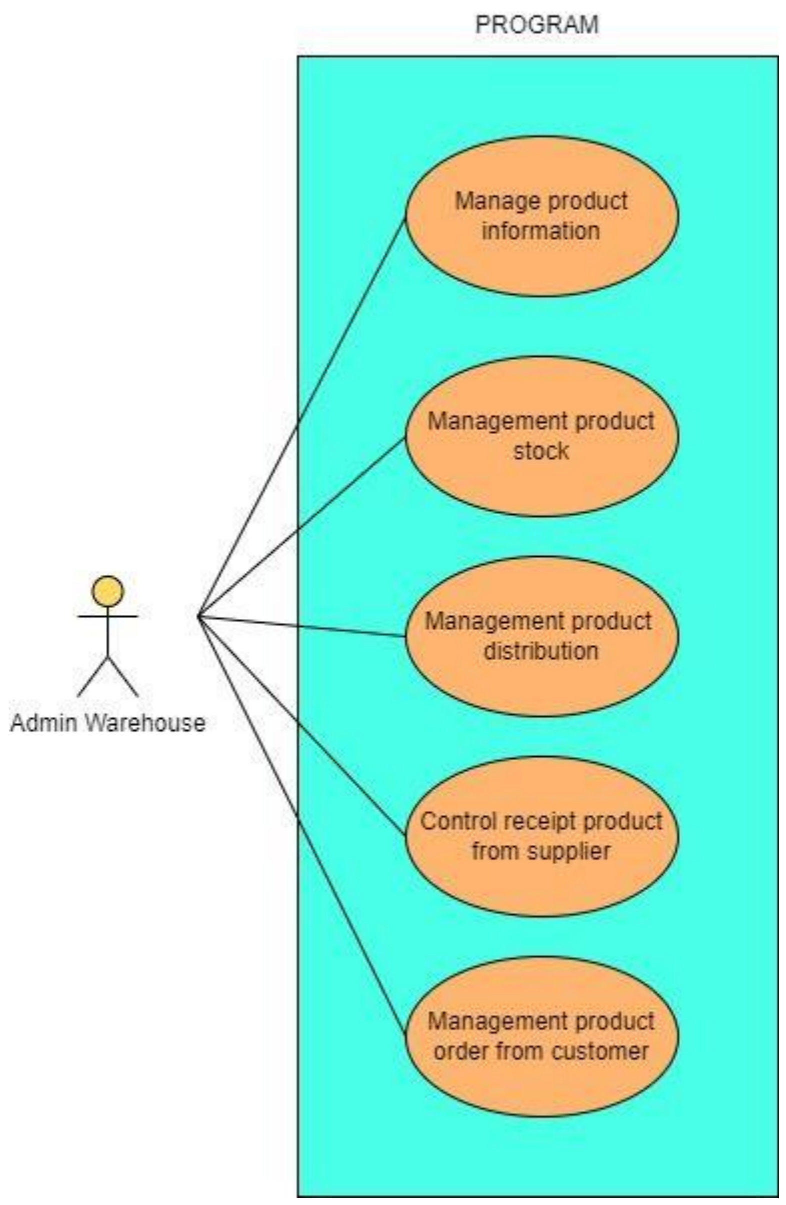
II. ER DIAGRAM

Here is our entity relational diagram that would explain how our program works.



III. USE CASE DIAGRAM

The following picture below is our Use Case Diagram


PROGRAM
Manage product information
Management product stock
Management product distribution
Control receipt product from supplier
Management product order from customer
Admin Warehouse

IV. DATABASE SCHEMA

5.1 Admin Table

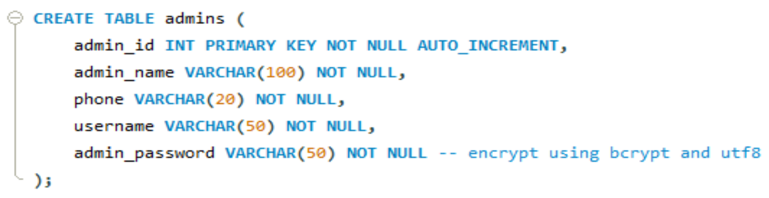
The "admins" table is used to manage and authenticate administrators who have access to the system or application. It allows for the creation, modification, and deletion of administrator accounts. The "username" and "admin_password" columns can be used for authentication purposes when an administrator logs into the system.

The "admins" table supports the following functionalities:

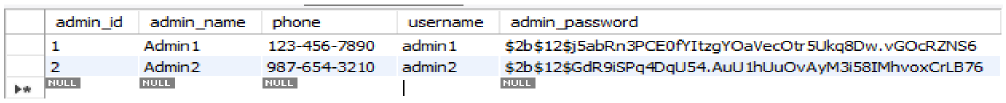
1. User Authentication: The table stores the usernames and encrypted passwords of administrators. When an administrator logs into the system, their credentials are checked against the stored values to authenticate their identity.
2. User Authorization: The table contains information about administrators' roles and privileges. By querying this table, the system can determine the level of access and permissions each administrator has within the system.
3. User Management: The table allows for the creation, modification, and deletion of administrator accounts. New administrators can be added to the system by inserting records into the "admins" table, while existing administrators can have their details updated or be removed from the table.
4. Profile Management: The table stores the personal information of administrators, such as their names and contact phone numbers. This information can be used for administrative purposes or to display user profiles within the system.
5. User Identification: The unique identifier (admin_id) assigned to each administrator in the table allows for easy referencing and identification of individual administrators throughout the system.

By querying the "admins" table, we can perform operations such as:

- Authenticating administrators based on their usernames and passwords.
- Retrieving information about specific administrators or a list of all administrators.
- Adding new administrators to the system.
- Modifying or updating administrator details, such as names or contact information.
- Removing administrators from the system.



```sql
CREATE TABLE admins (
    admin_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    admin_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    username VARCHAR(50) NOT NULL,
    admin_password VARCHAR(50) NOT NULL -- encrypt using bcrypt and utf8
);
```



	admin_id	admin_name	phone	username	admin_password
	1	Admin1	123-456-7890	admin1	$2b$12$j5abRn3PCE0fYItzgYOaVecOtr5Ukq8Dw.vGOcRZNS6
	2	Admin2	987-654-3210	admin2	$2b$12$GdR9iSPq4DqU54.AuU1hUuOvAyM3i58IMhvoxCrLB76
▶*	NULL	NULL	NULL		NULL

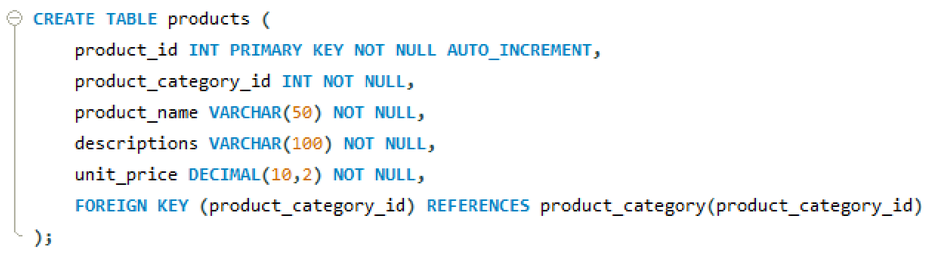
5.2 Product Table

The "products" table represents the products available in the database. It stores information about each product, including its unique identifier (product_id), the category it belongs to (product_category_id), the product name, descriptions, and unit price.

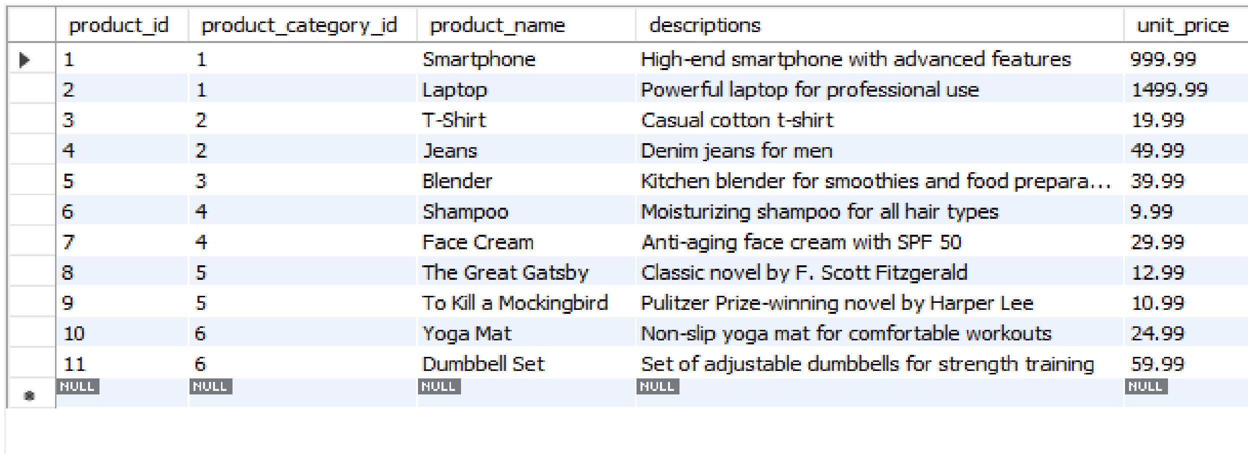
The function of the "products" table is to store and manage the data related to individual products. It serves as a central repository for product information and allows for the efficient retrieval, modification, and deletion of product records. The table establishes a relationship with the "product_category" table through the foreign key constraint on the "product_category_id" column, linking each product to its respective category.

By querying the "products" table, we can perform various operations such as:

- Retrieving a list of all products.
- Searching for a specific product by its name or ID.
- Obtaining detailed information about a product, including its category, descriptions, and price.
- Adding new products to the database.
- Updating product details such as descriptions or price.
- Removing products from the database.



```sql
CREATE TABLE products (
    product_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    product_category_id INT NOT NULL,
    product_name VARCHAR(50) NOT NULL,
    descriptions VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (product_category_id) REFERENCES product_category(product_category_id)
);
```



	product_id	product_category_id	product_name	descriptions	unit_price
▶	1	1	Smartphone	High-end smartphone with advanced features	999.99
	2	1	Laptop	Powerful laptop for professional use	1499.99
	3	2	T-Shirt	Casual cotton t-shirt	19.99
	4	2	Jeans	Denim jeans for men	49.99
	5	3	Blender	Kitchen blender for smoothies and food prepara...	39.99
	6	4	Shampoo	Moisturizing shampoo for all hair types	9.99
	7	4	Face Cream	Anti-aging face cream with SPF 50	29.99
	8	5	The Great Gatsby	Classic novel by F. Scott Fitzgerald	12.99
	9	5	To Kill a Mockingbird	Pulitzer Prize-winning novel by Harper Lee	10.99
	10	6	Yoga Mat	Non-slip yoga mat for comfortable workouts	24.99
	11	6	Dumbbell Set	Set of adjustable dumbbells for strength training	59.99
*	NULL	NULL	NULL	NULL	NULL

5.3 Product Category Table

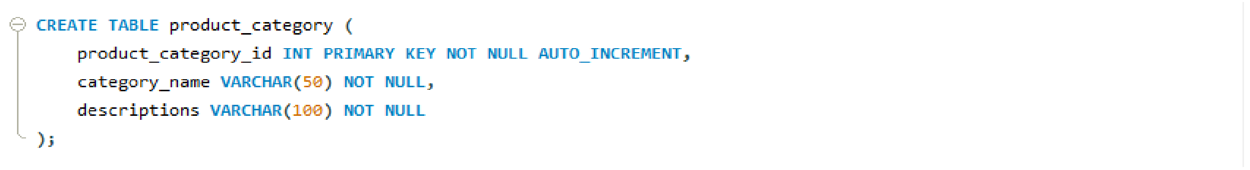
The function of the "product_category" table is to classify and categorize products based on their types or genres. It serves as a reference table that holds the distinct categories available in the system.

The "product_category" table allows for efficient management and retrieval of product categories. It can be used for various purposes, including:

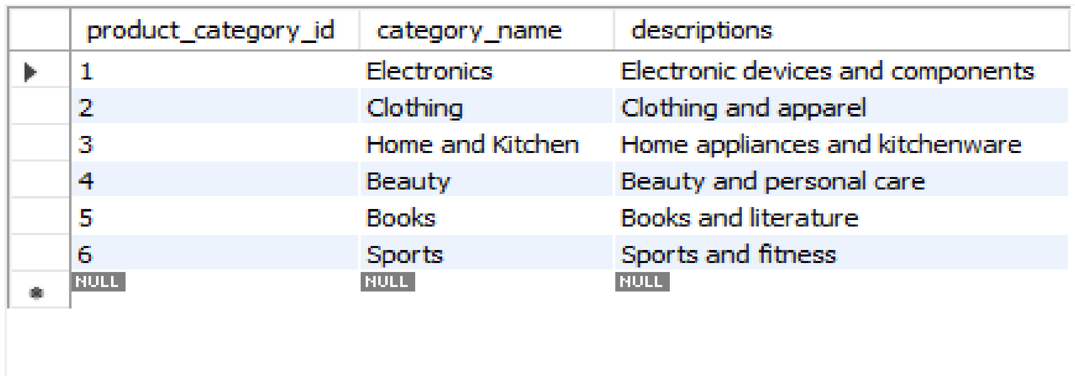
- Categorizing products and organizing them into meaningful groups.
- Providing a predefined set of categories for users to choose from when adding or updating products.
- Enabling filtering or sorting of products based on their category.
- Displaying a list of available categories for users to browse and select.
- Establishing a relationship with the "products" table through the foreign key constraint on the "product_category_id" column, enabling the association of products with their respective categories.

By querying the "product_category" table, we can perform operations such as:

- Retrieving a list of all product categories.
- Searching for a specific category by its name or ID.
- Adding new categories to the database.
- Modifying or updating category names or descriptions.
- Removing categories from the database (if required).



```
CREATE TABLE product_category (
    product_category_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    category_name VARCHAR(50) NOT NULL,
    descriptions VARCHAR(100) NOT NULL
);
```



product_category_id	category_name	descriptions
1	Electronics	Electronic devices and components
2	Clothing	Clothing and apparel
3	Home and Kitchen	Home appliances and kitchenware
4	Beauty	Beauty and personal care
5	Books	Books and literature
6	Sports	Sports and fitness
NULL	NULL	NULL

5.4 Product Stock Table

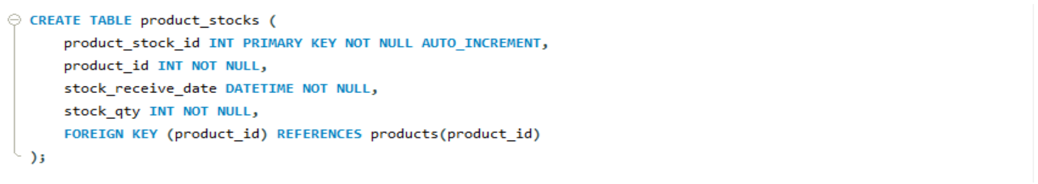
The function of the "product_stocks" table is to track and manage the stock levels of individual products. It allows for the recording of stock movements, such as receiving new stock or selling existing stock, and provides information about the current stock status.

The "product_stocks" table supports the following functionalities:

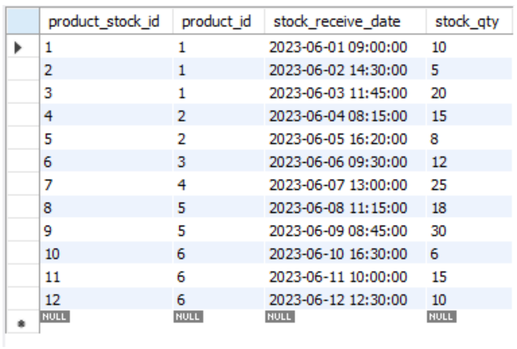
1. Stock Management: The table keeps track of the quantity of each product available in the warehouse. By updating the stock quantity in this table, you can reflect changes in the inventory levels based on stock receipts or sales.
2. Stock Receiving: The table records the receive date and quantity when new stock for a product is received in the warehouse. This allows for tracking the history of stock receipts and facilitates stock management.
3. Stock Reporting: By querying the "product_stocks" table, you can generate reports on the current stock levels for different products. These reports help in monitoring inventory, identifying low stock levels, and making informed decisions regarding restocking.
4. Stock Availability: The table provides information about the availability of each product by storing the quantity of stock. This allows for checking the availability of a particular product before processing orders or making sales.
5. Stock History: The table maintains a historical record of stock movements, including stock receipts. This enables tracking the timeline and quantity of stock received for each product, providing a complete audit trail.

By querying the "product_stocks" table, we can perform operations such as:

- Retrieving the current stock levels for a specific product.
- Tracking the stock history of a product, including stock receipt dates and quantities.
- Updating the stock quantity when new stock is received or when stock is sold.
- Generating reports on stock availability and identifying low stock levels.



```
CREATE TABLE product_stocks (
    product_stock_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    product_id INT NOT NULL,
    stock_receive_date DATETIME NOT NULL,
    stock_qty INT NOT NULL,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```



product_stock_id	product_id	stock_receive_date	stock_qty
1	1	2023-06-01 09:00:00	10
2	1	2023-06-02 14:30:00	5
3	1	2023-06-03 11:45:00	20
4	2	2023-06-04 08:15:00	15
5	2	2023-06-05 16:20:00	8
6	3	2023-06-06 09:30:00	12
7	4	2023-06-07 13:00:00	25
8	5	2023-06-08 11:15:00	18
9	5	2023-06-09 08:45:00	30
10	6	2023-06-10 16:30:00	6
11	6	2023-06-11 10:00:00	15
12	6	2023-06-12 12:30:00	10
NULL	NULL	NULL	NULL

5.5 Supplier Table

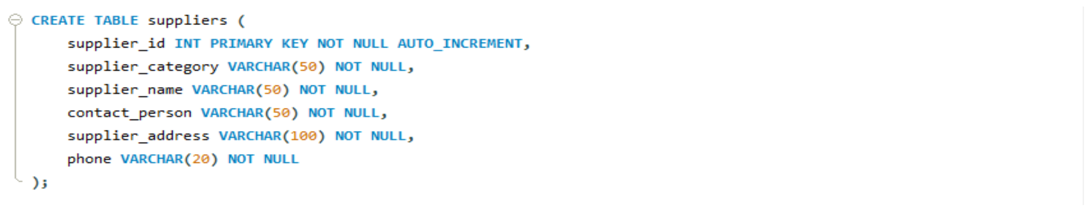
The function of the "suppliers" table is to manage and store information about the suppliers the company engages with. It enables the company to maintain a database of suppliers and their relevant details for efficient procurement and communication purposes.

The "suppliers" table supports the following functionalities:

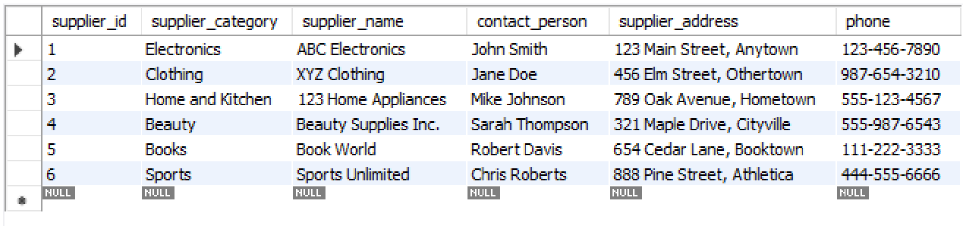
1. Supplier Management: The table allows for the creation, modification, and deletion of supplier records. New suppliers can be added to the system by inserting records into the "suppliers" table, while existing supplier details can be updated or removed as necessary.
2. Supplier Information: The table stores important information about suppliers, including their names, contact persons, addresses, and phone numbers. This data can be used for communication, vendor evaluation, and supplier relationship management.
3. Categorization: The table includes a column for supplier category, which allows suppliers to be classified based on their specialization or the type of products they provide. This categorization can assist in filtering or sorting suppliers based on specific criteria.
4. Supplier Relationships: The table establishes relationships between suppliers and other tables in the database, such as the "products" table. By using the foreign key constraint on the "supplier_id" column, the table can associate products with their respective suppliers, facilitating supplier-product mapping and enabling efficient querying of product-supplier information.
5. Vendor Selection: The table provides a list of available suppliers, which can be used during the vendor selection process. The information stored in the table allows the company to evaluate and compare different suppliers based on their attributes.

By querying the "suppliers" table, we can perform operations such as:

- Retrieving a list of all suppliers or searching for specific suppliers based on their names or categories.
- Obtaining detailed information about a particular supplier, including contact details and address.
- Adding new suppliers to the system or updating existing supplier records.
- Associating suppliers with specific products or transactions through the use of foreign keys.



```
CREATE TABLE suppliers (
    supplier_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    supplier_category VARCHAR(50) NOT NULL,
    supplier_name VARCHAR(50) NOT NULL,
    contact_person VARCHAR(50) NOT NULL,
    supplier_address VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL
);
```



supplier_id	supplier_category	supplier_name	contact_person	supplier_address	phone
1	Electronics	ABC Electronics	John Smith	123 Main Street, Anytown	123-456-7890
2	Clothing	XYZ Clothing	Jane Doe	456 Elm Street, Othertown	987-654-3210
3	Home and Kitchen	123 Home Appliances	Mike Johnson	789 Oak Avenue, Hometown	555-123-4567
4	Beauty	Beauty Supplies Inc.	Sarah Thompson	321 Maple Drive, Cityville	555-987-6543
5	Books	Book World	Robert Davis	654 Cedar Lane, Booktown	111-222-3333
6	Sports	Sports Unlimited	Chris Roberts	888 Pine Street, Athletica	444-555-6666
NULL	NULL	NULL	NULL	NULL	NULL

5.6 Receipt Table

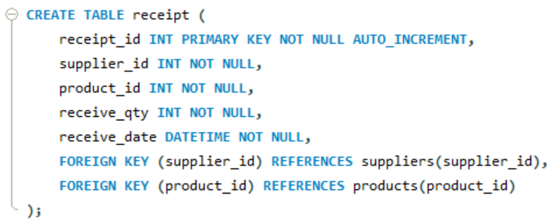
The function of the "receipt" table is to track and record the details of product deliveries from suppliers. It allows for the documentation of received quantities, dates, and the association of products with their respective suppliers.

The "receipt" table supports the following functionalities:

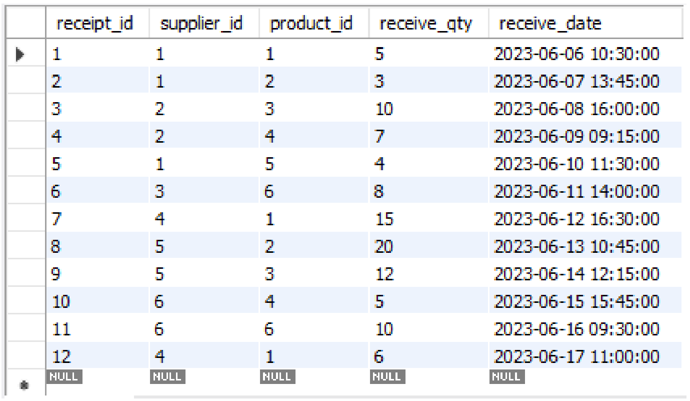
1. Receiving Product Quantities: The table records the quantities of products received from suppliers. This information helps in tracking and updating the inventory levels based on the received quantities.
2. Supplier Management: The table associates each receipt with a specific supplier through the supplier_id column. This allows for tracking which supplier delivered the products for each receipt and facilitates supplier performance evaluation and analysis.
3. Stock Updates: By recording the quantities received in the "receipt" table, the stock levels in the "product_stocks" table can be updated accordingly. This ensures that the inventory accurately reflects the received products and allows for maintaining accurate stock information.
4. Audit Trail: The table maintains a historical record of receipts, including the date and time of each receipt. This creates an audit trail of product deliveries and provides a reference for future inquiries or investigations.

By querying the "receipt" table, we can perform operations such as:

- Retrieving information about specific receipts, including the supplier, product, quantity received, and receipt date.
- Generating reports on product deliveries from suppliers within a specific time period.
- Analyzing supplier performance based on the number and frequency of receipts.
- Updating stock quantities in the "product_stocks" table based on the quantities received in the receipts.
- Identifying the latest product deliveries and the associated suppliers.



```
CREATE TABLE receipt (
    receipt_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    supplier_id INT NOT NULL,
    product_id INT NOT NULL,
    receive_qty INT NOT NULL,
    receive_date DATETIME NOT NULL,
    FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```



receipt_id	supplier_id	product_id	receive_qty	receive_date
1	1	1	5	2023-06-06 10:30:00
2	1	2	3	2023-06-07 13:45:00
3	2	3	10	2023-06-08 16:00:00
4	2	4	7	2023-06-09 09:15:00
5	1	5	4	2023-06-10 11:30:00
6	3	6	8	2023-06-11 14:00:00
7	4	1	15	2023-06-12 16:30:00
8	5	2	20	2023-06-13 10:45:00
9	5	3	12	2023-06-14 12:15:00
10	6	4	5	2023-06-15 15:45:00
11	6	6	10	2023-06-16 09:30:00
12	4	1	6	2023-06-17 11:00:00
NULL	NULL	NULL	NULL	NULL

5.7 Customer Table

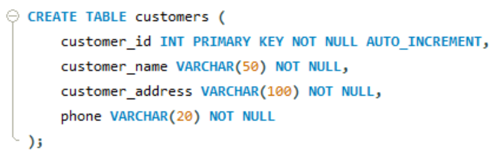
The function of the "customers" table is to manage and store information about the customers who interact with the company. It allows for the creation of customer records, maintenance of customer details, and tracking of customer-related information for various business processes.

The "customers" table supports the following functionalities:

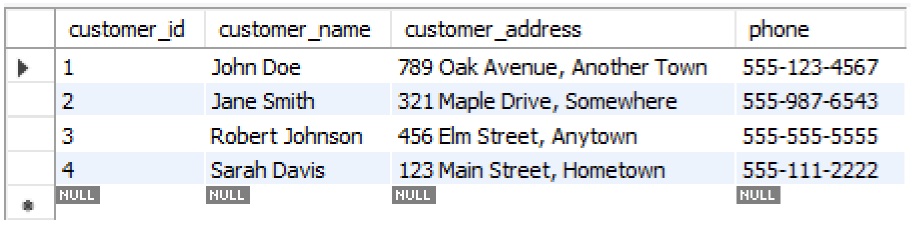
1. Customer Management: The table allows for the creation, modification, and deletion of customer records. New customers can be added to the system by inserting records into the "customers" table, while existing customer details can be updated or removed as necessary.
2. Customer Information: The table stores important information about customers, including their names, addresses, and contact details. This data can be used for communication, order fulfillment, and customer relationship management.
3. Customer Order Tracking: By associating the "customers" table with the "orders" table through the customer_id foreign key, the table allows for tracking and querying customer-specific orders. This enables the retrieval of order history and facilitates order management and tracking for individual customers.
4. Customer Contact Information: The table stores customer contact details, such as addresses and phone numbers. This information can be utilized for order delivery, communication, and customer support purposes.
5. Customer Reporting: By querying the "customers" table, you can generate reports and insights on customer demographics, sales patterns, and customer behavior. This information can be utilized for targeted marketing campaigns, customer segmentation, and business decision-making.

By querying the "customers" table, we can perform operations such as:

- Retrieving a list of all customers or searching for specific customers based on their names or addresses.
- Obtaining detailed information about a particular customer, including contact details and order history.
- Adding new customers to the system or updating existing customer records.
- Associating customers with their respective orders through the use of foreign keys.



```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    customer_name VARCHAR(50) NOT NULL,
    customer_address VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL
);
```



	customer_id	customer_name	customer_address	phone
▶	1	John Doe	789 Oak Avenue, Another Town	555-123-4567
	2	Jane Smith	321 Maple Drive, Somewhere	555-987-6543
	3	Robert Johnson	456 Elm Street, Anytown	555-555-5555
	4	Sarah Davis	123 Main Street, Hometown	555-111-2222
*	NULL	NULL	NULL	NULL

5.8 Order Table

The function of the "orders" table is to track and record customer orders. It allows for the documentation of the products ordered, quantities, associated customers, and order dates.

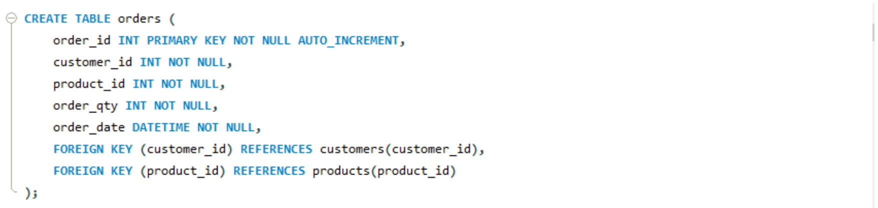
The "orders" table supports the following functionalities:

1. Order Placement: The table records the details of orders placed by customers. This includes the specific products ordered, the quantity of each product, and the associated customer. This information helps in tracking and fulfilling customer orders.
2. Customer Relationship Management: By associating each order with a specific customer through the customer_id column, the table facilitates customer relationship management. It enables tracking and analyzing customer order history, preferences, and purchasing behavior.
3. Inventory Management: The table captures the product_id and order_qty, which helps in managing and updating the inventory levels. By deducting the ordered quantity from the available stock, the table assists in maintaining accurate stock information and avoids overselling.
4. Order Fulfillment: The table stores the date and time of each order (order_date), allowing for the tracking of order processing and fulfillment. This information can be used to prioritize and schedule order processing, manage delivery timelines, and provide updates to customers regarding their orders.
5. Reporting and Analysis: By querying the "orders" table, you can generate reports and analyze various aspects of the orders, such as sales volume, popular products, customer

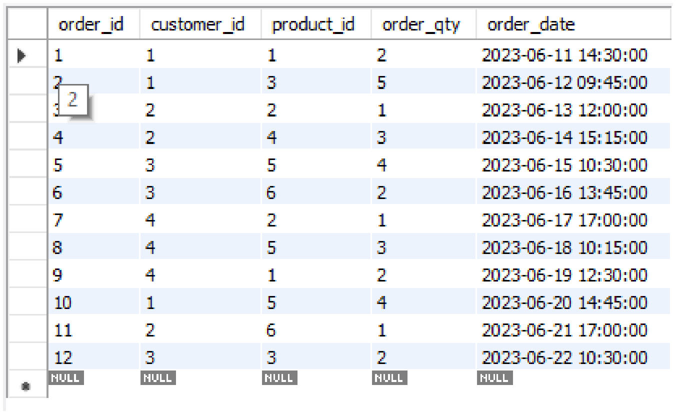
behavior, and order trends. This information can be utilized for business analysis, forecasting, and decision-making.

By querying the "orders" table, we can perform operations such as:

- Retrieving information about specific orders, including the customer, product, quantity, and order date.
- Generating reports on order history, such as orders within a specific time period or orders associated with a particular customer.
- Analyzing sales patterns, product popularity, and customer purchasing behavior.
- Updating stock quantities in the "product_stocks" table based on the quantities ordered.
- Tracking the progress and status of order fulfillment.



```
CREATE TABLE orders (
    order_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    customer_id INT NOT NULL,
    product_id INT NOT NULL,
    order_qty INT NOT NULL,
    order_date DATETIME NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```



order_id	customer_id	product_id	order_qty	order_date
1	1	1	2	2023-06-11 14:30:00
2	1	3	5	2023-06-12 09:45:00
3	2	2	1	2023-06-13 12:00:00
4	2	4	3	2023-06-14 15:15:00
5	3	5	4	2023-06-15 10:30:00
6	3	6	2	2023-06-16 13:45:00
7	4	2	1	2023-06-17 17:00:00
8	4	5	3	2023-06-18 10:15:00
9	4	1	2	2023-06-19 12:30:00
10	1	5	4	2023-06-20 14:45:00
11	2	6	1	2023-06-21 17:00:00
12	3	3	2	2023-06-22 10:30:00
NULL	NULL	NULL	NULL	NULL

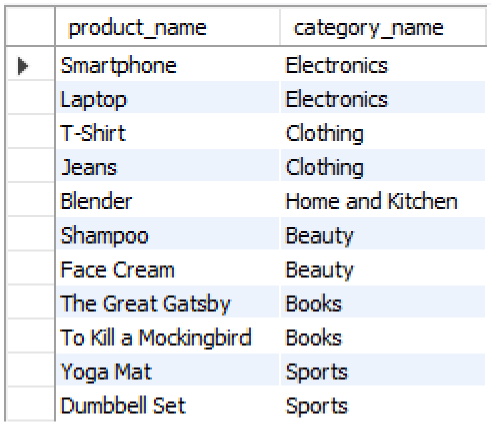
V. POPULATING DATABASE

```
-- INSERT INTO product_category
INSERT INTO product_category (category_name, descriptions) VALUES
('Electronics', 'Electronic devices and components'),
('Clothing', 'Clothing and apparel'),
('Home and Kitchen', 'Home appliances and kitchenware'),
('Beauty', 'Beauty and personal care'),
('Books', 'Books and literature'),
('Sports', 'Sports and fitness');

-- INSERT INTO products
INSERT INTO products (product_category_id, product_name, descriptions,
unit_price) VALUES
(1, 'Smartphone', 'High-end smartphone with advanced features', 999.99),
(1, 'Laptop', 'Powerful laptop for professional use', 1499.99),
(2, 'T-Shirt', 'Casual cotton t-shirt', 19.99),
(2, 'Jeans', 'Denim jeans for men', 49.99),
(3, 'Blender', 'Kitchen blender for smoothies and food preparation',
39.99),
(4, 'Shampoo', 'Moisturizing shampoo for all hair types', 9.99),
(4, 'Face Cream', 'Anti-aging face cream with SPF 50', 29.99),
(5, 'The Great Gatsby', 'Classic novel by F. Scott Fitzgerald', 12.99),
(5, 'To Kill a Mockingbird', 'Pulitzer Prize-winning novel by Harper Lee',
10.99),
(6, 'Yoga Mat', 'Non-slip yoga mat for comfortable workouts', 24.99),
(6, 'Dumbbell Set', 'Set of adjustable dumbbells for strength training',
59.99);

-- INSERT INTO product_stocks
INSERT INTO product_stocks (product_id, stock_receive_date, stock_qty)
VALUES
(1, '2023-06-01 09:00:00', 10),
(1, '2023-06-02 14:30:00', 5),
(1, '2023-06-03 11:45:00', 20),
(2, '2023-06-04 08:15:00', 15),
(2, '2023-06-05 16:20:00', 8),
(3, '2023-06-06 09:30:00', 12),
(4, '2023-06-07 13:00:00', 25),
(5, '2023-06-08 11:15:00', 18),
(5, '2023-06-09 08:45:00', 30),
(6, '2023-06-10 16:30:00', 6),
(6, '2023-06-11 10:00:00', 15),
(6, '2023-06-12 12:30:00', 10);


-- INSERT INTO suppliers
INSERT INTO suppliers (supplier_category, supplier_name, contact_person,
supplier_address, phone) VALUES
('Electronics', 'ABC Electronics', 'John Smith', '123 Main Street,
Anytown', '123-456-7890'),
('Clothing', 'XYZ Clothing', 'Jane Doe', '456 Elm Street, Othertown',
'987-654-3210'),
('Home and Kitchen', '123 Home Appliances', 'Mike Johnson', '789 Oak
Avenue, Hometown', '555-123-4567'),
('Beauty', 'Beauty Supplies Inc.', 'Sarah Thompson', '321 Maple Drive,
Cityville', '555-987-6543'),
('Books', 'Book World', 'Robert Davis', '654 Cedar Lane, Booktown', '111-
222-3333'),
('Sports', 'Sports Unlimited', 'Chris Roberts', '888 Pine Street,
Athletica', '444-555-6666');

-- INSERT INTO receipt
INSERT INTO receipt (supplier_id, product_id, receive_qty, receive_date)
VALUES
(1, 1, 5, '2023-06-06 10:30:00'),
(1, 2, 3, '2023-06-07 13:45:00'),
(2, 3, 10, '2023-06-08 16:00:00'),
(2, 4, 7, '2023-06-09 09:15:00'),
(1, 5, 4, '2023-06-10 11:30:00'),
(3, 6, 8, '2023-06-11 14:00:00'),
(4, 1, 15, '2023-06-12 16:30:00'),
(5, 2, 20, '2023-06-13 10:45:00'),
(5, 3, 12, '2023-06-14 12:15:00'),
(6, 4, 5, '2023-06-15 15:45:00'),
(6, 6, 10, '2023-06-16 09:30:00'),
(4, 1, 6, '2023-06-17 11:00:00');


-- INSERT INTO admins
INSERT INTO admins (admin_id, admin_name, phone, username, admin_password)
VALUES
-- real password for admin1 is "password1" it's has been encrypt using
bcrypt gensalt and hash with utf8
(1, 'Admin1', '123-456-7890', 'admin1',
'$2b$12$j5abRn3PCE0fYItzgYOaVecOtr5Ukq8Dw.vGOcRZNS6DdO9j6/2Lq'),
-- real password for admin2 is "password2" it's has been encrypt using
bcrypt gensalt and hash with utf8
(2, 'Admin2', '987-654-3210', 'admin2',
'$2b$12$GdR9iSPq4DqU54.AuU1hUuOvAyM3i58IMhvoxCrLB76v.3tnk.RNe');

-- INSERT INTO customers
INSERT INTO customers (customer_name, customer_address, phone) VALUES
('John Doe', '789 Oak Avenue, Another Town', '555-123-4567'),
('Jane Smith', '321 Maple Drive, Somewhere', '555-987-6543'),
('Robert Johnson', '456 Elm Street, Anytown', '555-555-5555'),
('Sarah Davis', '123 Main Street, Hometown', '555-111-2222');

-- INSERT INTO orders
INSERT INTO orders (customer_id, product_id, order_qty, order_date) VALUES
(1, 1, 2, '2023-06-11 14:30:00'),
(1, 3, 5, '2023-06-12 09:45:00'),
(2, 2, 1, '2023-06-13 12:00:00'),
(2, 4, 3, '2023-06-14 15:15:00'),
(3, 5, 4, '2023-06-15 10:30:00'),
(3, 6, 2, '2023-06-16 13:45:00'),
(4, 2, 1, '2023-06-17 17:00:00'),
(4, 5, 3, '2023-06-18 10:15:00'),
(4, 1, 2, '2023-06-19 12:30:00'),
(1, 5, 4, '2023-06-20 14:45:00'),
(2, 6, 1, '2023-06-21 17:00:00'),
(3, 3, 2, '2023-06-22 10:30:00');
```

VI. QUERIES

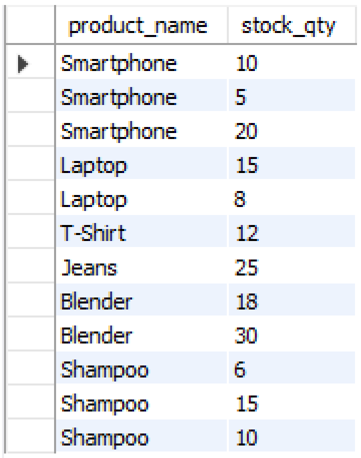
6.1 Product Operations

```
-- Display all product and related category
SELECT p.product_name, c.category_name
FROM products p
JOIN product_category c ON p.product_category_id = c.product_category_id;
```



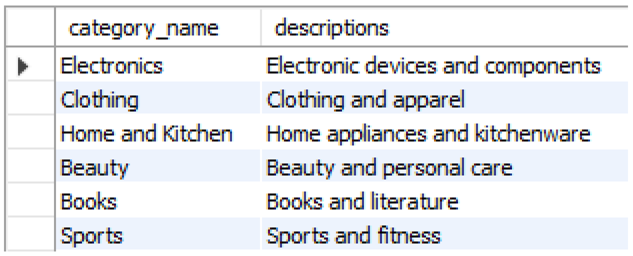
product_name	category_name
Smartphone	Electronics
Laptop	Electronics
T-Shirt	Clothing
Jeans	Clothing
Blender	Home and Kitchen
Shampoo	Beauty
Face Cream	Beauty
The Great Gatsby	Books
To Kill a Mockingbird	Books
Yoga Mat	Sports
Dumbbell Set	Sports

```
-- Display product with available stock
SELECT p.product_name, ps.stock_qty
FROM products p
JOIN product_stocks ps ON p.product_id = ps.product_id
WHERE ps.stock_qty > 0;
```

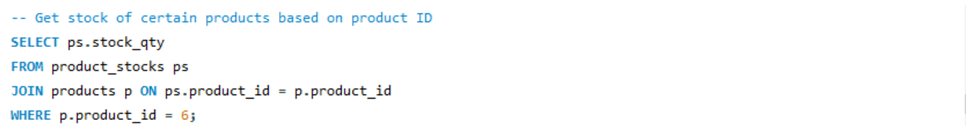


product_name	stock_qty
Smartphone	10
Smartphone	5
Smartphone	20
Laptop	15
Laptop	8
T-Shirt	12
Jeans	25
Blender	18
Blender	30
Shampoo	6
Shampoo	15
Shampoo	10

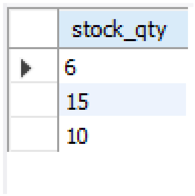
```
-- Get a list of all product categories along with their descriptions
SELECT category_name, descriptions FROM product_category;
```



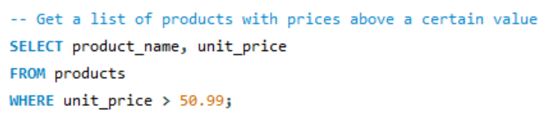
	category_name	descriptions
▶	Electronics	Electronic devices and components
	Clothing	Clothing and apparel
	Home and Kitchen	Home appliances and kitchenware
	Beauty	Beauty and personal care
	Books	Books and literature
	Sports	Sports and fitness



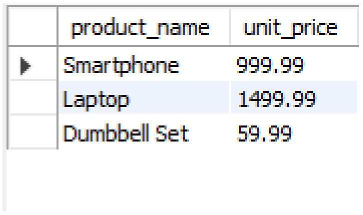
```sql
-- Get stock of certain products based on product ID
SELECT ps.stock_qty
FROM product_stocks ps
JOIN products p ON ps.product_id = p.product_id
WHERE p.product_id = 6;
```



	stock_qty
▶	6
	15
	10

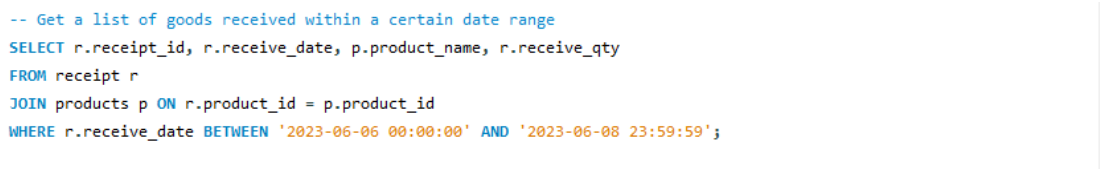


```sql
-- Get a list of products with prices above a certain value
SELECT product_name, unit_price
FROM products
WHERE unit_price > 50.99;
```

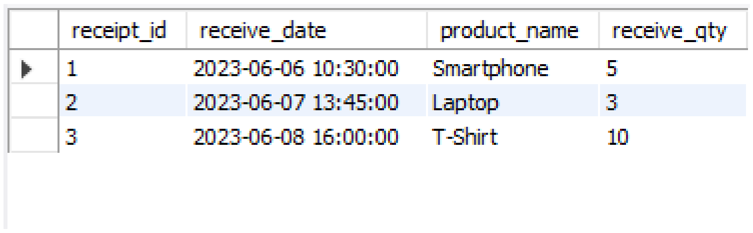


	product_name	unit_price
▶	Smartphone	999.99
	Laptop	1499.99
	Dumbbell Set	59.99

6.2 Receipt Operation

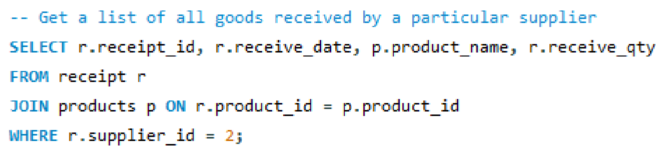


```sql
-- Get a list of goods received within a certain date range
SELECT r.receipt_id, r.receive_date, p.product_name, r.receive_qty
FROM receipt r
JOIN products p ON r.product_id = p.product_id
WHERE r.receive_date BETWEEN '2023-06-06 00:00:00' AND '2023-06-08 23:59:59';
```

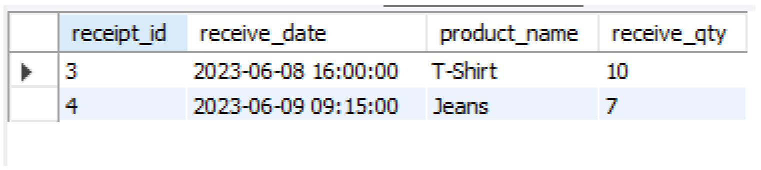


	receipt_id	receive_date	product_name	receive_qty
▶	1	2023-06-06 10:30:00	Smartphone	5
	2	2023-06-07 13:45:00	Laptop	3
	3	2023-06-08 16:00:00	T-Shirt	10

6.3 Supplier Operations

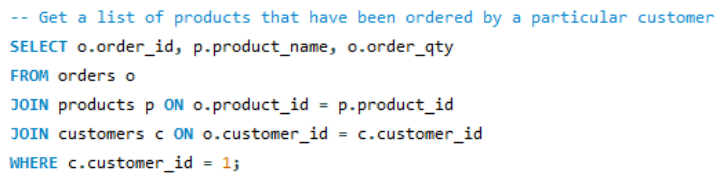


```
-- Get a list of all goods received by a particular supplier
SELECT r.receipt_id, r.receive_date, p.product_name, r.receive_qty
FROM receipt r
JOIN products p ON r.product_id = p.product_id
WHERE r.supplier_id = 2;
```

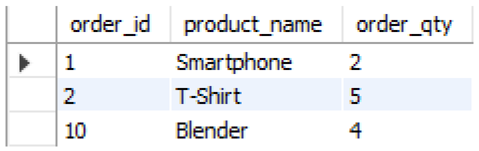


receipt_id	receive_date	product_name	receive_qty
3	2023-06-08 16:00:00	T-Shirt	10
4	2023-06-09 09:15:00	Jeans	7

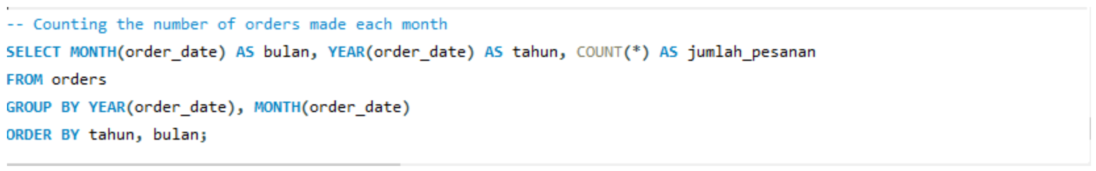
6.4 Order Operations



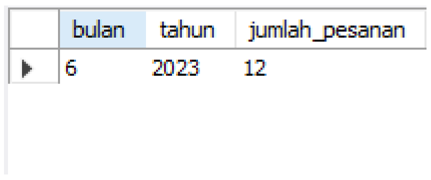
```
-- Get a list of products that have been ordered by a particular customer
SELECT o.order_id, p.product_name, o.order_qty
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN customers c ON o.customer_id = c.customer_id
WHERE c.customer_id = 1;
```



order_id	product_name	order_qty
1	Smartphone	2
2	T-Shirt	5
10	Blender	4



```
-- Counting the number of orders made each month
SELECT MONTH(order_date) AS bulan, YEAR(order_date) AS tahun, COUNT(*) AS jumlah_pesanan
FROM orders
GROUP BY YEAR(order_date), MONTH(order_date)
ORDER BY tahun, bulan;
```



bulan	tahun	jumlah_pesanan
6	2023	12