# Project Report

# Relational Database Design for E-Commerce Management

**for**

# INFO6210 - Data Management and Database Design

**Submitted By**

**Gangrade, Mayank(001837520)**

**SEC 06 - Fall 2017**

# 1. Problem Statement

The goal of this project will be to design a relational database that will describe all the entity related to commerce and properly covers all the relative scenarios in an e-commerce business using relational database management systems(RDBMS) like MySQL, MS SQL Server, OracleDB etc.

# 2. Design Approach

MySQL Workbench tool is used for designing database for an e-commerce business. Which can be able to track the record of Customers, Suppliers, Products, Customer orders, Customer payments and order delivery. A customer places an order for purchasing product which is supplied by a supplier. A customer can place an order for multiple products and a particular product can be supplied by multiple suppliers.

Entities are created for capturing data for customers, suppliers, orders, and payment used by customers.

Finally, views, triggers, procedures are written for applying run-time conditions and performing analytics.

# 3. Entities of concern

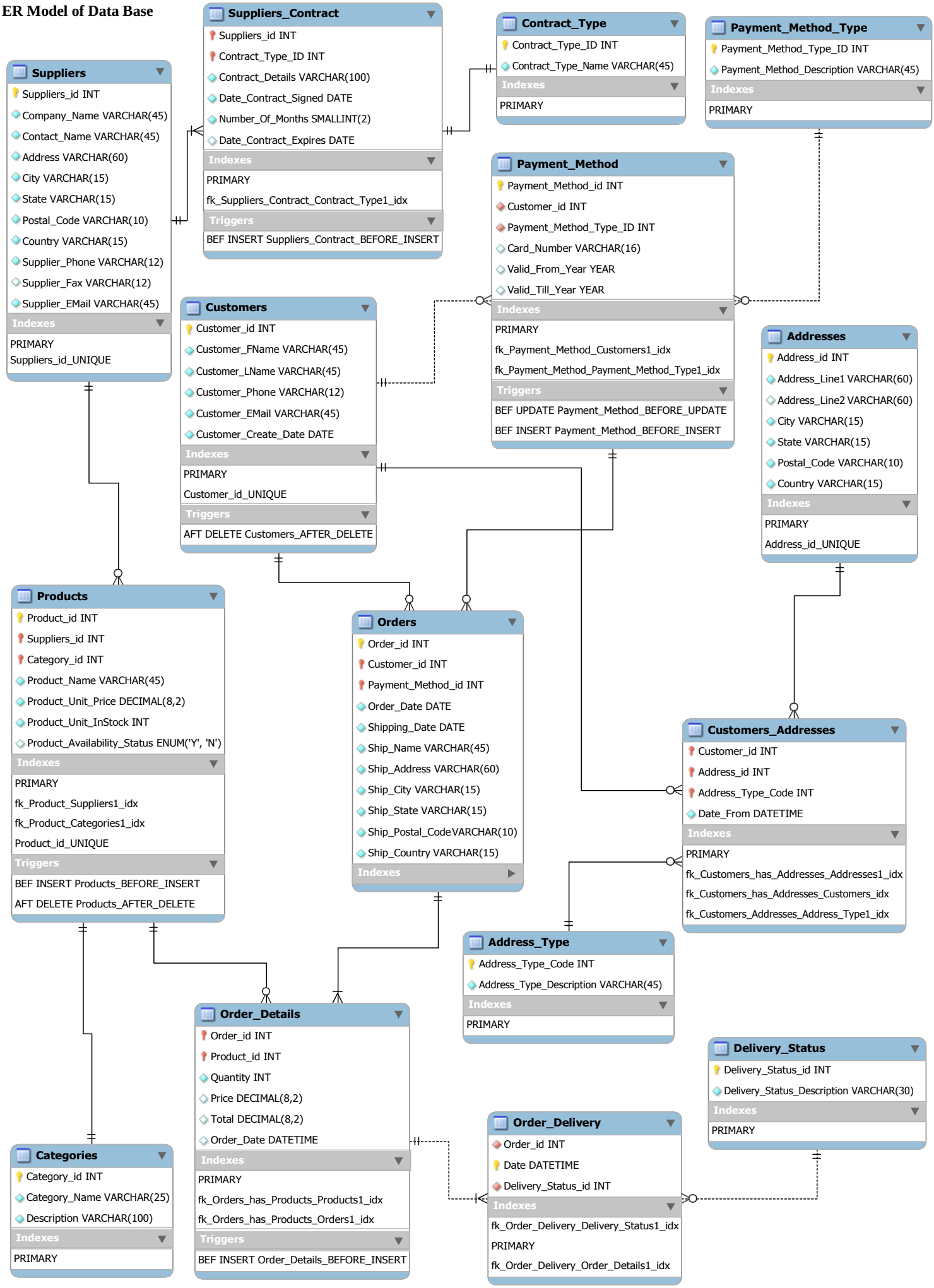
**Customer:** Details of the customer like the First name, Last name, Address details, Contact details, payment method used by customer to place order

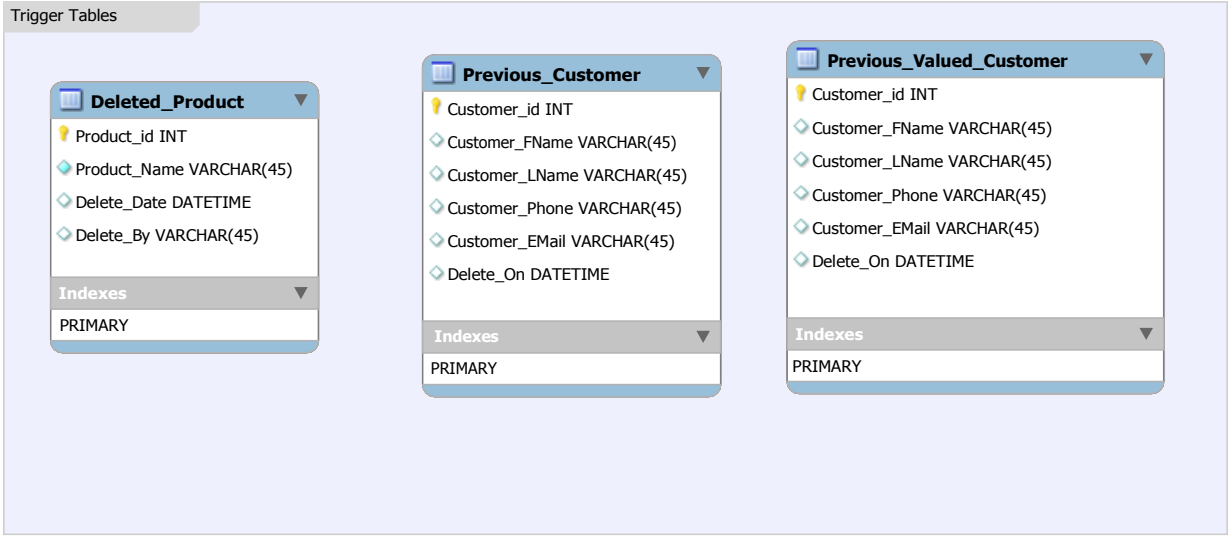
**Supplier:** Details of the supplier like Name, Contact person name, contact details, Address details, contract details

**Product:** Details of the product like Product name, Product Category, Price, availability

**Order:** Details of the order placed by customer like Product, Quantity of product, order date, Order delivery status

# ER Model of Data Base




Trigger Tables
Deleted_Product
Product_id INT
Product_Name VARCHAR(45)
Delete_Date DATETIME
Delete_By VARCHAR(45)
Indexes
PRIMARY
Previous_Customer
Customer_id INT
Customer_FName VARCHAR(45)
Customer_LName VARCHAR(45)
Customer_Phone VARCHAR(45)
Customer_EMail VARCHAR(45)
Delete_On DATETIME
Indexes
PRIMARY
Previous_Valued_Customer
Customer_id INT
Customer_FName VARCHAR(45)
Customer_LName VARCHAR(45)
Customer_Phone VARCHAR(45)
Customer_EMail VARCHAR(45)
Delete_On DATETIME
Indexes
PRIMARY

**View :**


Customer_Order_Prodcut_Supplier_Details

# 4. Triggers

*"Trigger is a database object that is associated with a table. It will be activated when a defined action is executed for the table. The trigger can be executed when you run one of the following statements on the table: INSERT, UPDATE and DELETE and it can be invoked before or after the event."*

Created triggers:

**1. Customers_AFTER_DELETE:** A customer who orders if any product becomes valued customer so when a customer record is deleted it stores in **Previous_Valued_Customer** table else in **Previous_Customer** table.

```
CREATE DEFINER = CURRENT_USER
TRIGGER `AmazonDB`.`Customers_AFTER_DELETE`
AFTER DELETE ON `Customers` FOR EACH ROW
BEGIN
    DECLARE Placed_Order_Count INT;
    SET @Placed_Order_Count = 0;
SELECT
    COUNT(*)
INTO Placed_Order_Count FROM
    Orders o
WHERE
    o.Customer_id = OLD.Customer_id;
    IF Placed_Order_Count > 0 THEN
        INSERT INTO Previous_Valued_Customer(Customer_id, Customer_FName, Customer_LName,Customer_Phone, Customer_EMail, Delete_On)
        Values (old.Customer_id, old.Customer_FName, old.Customer_LName, old.Customer_Phone, old.Customer_EMail, now());

    ELSE
        INSERT INTO Previous_Customer(Customer_id, Customer_FName, Customer_LName,Customer_Phone, Customer_EMail, Delete_On)
        Values (old.Customer_id, old.Customer_FName, old.Customer_LName, old.Customer_Phone, old.Customer_EMail, now());
    END IF;

    -- Clean up orders table
DELETE FROM Orders
WHERE
    Orders.Customer_id = old.Customer_id;

END
```

**2. Order_Details_BEFORE_INSERT:** Used to verify number of unit available for a particular product. If ordered product unit is less than or equals to available product unit in stock then order successfully places and same quantity reduced at the same time from Products table.

```
CREATE DEFINER = CURRENT_USER
TRIGGER `AmazonDB`.`Order_Details_BEFORE_INSERT`
BEFORE INSERT ON `Order_Details` FOR EACH ROW
BEGIN
    declare quantity INTEGER;
    declare price DECIMAL(8,2);

    SET @quantity = (Select p.Product_Unit_InStock From Products p WHERE p.Product_id = NEW.Product_id);

    IF NEW.Quantity > @quantity THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Quantity not available';

    ELSEIF NEW.Quantity <= @quantity THEN

    UPDATE Products
    SET Product_Unit_InStock = @quantity - NEW.Quantity
    WHERE Products.Product_id = NEW.Product_id;

    SET @price = (Select p.Product_Unit_Price From Products p WHERE p.Product_id = NEW.Product_id);

    SET NEW.Price = @price;
    SET NEW.Total = NEW.Quantity * @price;
    SET NEW.Order_Date = now();
    END IF;

END
```

**3. Suppliers_Contract_BEFORE_INSERT:** Used to calculate contract expiration date on the basis of contract start date and number of months.

```
CREATE DEFINER = CURRENT_USER
TRIGGER `AmazonDB`.`Suppliers_Contract_BEFORE_INSERT`
BEFORE INSERT ON `Suppliers_Contract` FOR EACH ROW
BEGIN
    SET
    NEW.Date_Contract_Expires =
    DATE_ADD(NEW.Date_Contract_Signed, INTERVAL NEW.Number_Of_Months MONTH);
END
```

# 5. Stored Procedure

*"A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs."*

Created stored procedure:

**1. SP_Order_Details_Retrival:** Used to display all the orders placed by customers by taking Order_ID and Cust_ID as input.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SP_Order_Details_Retrival`(IN Order_ID INT, IN Cust_ID int)
BEGIN
    SELECT
    orders.Order_id,
    orders.Customer_id,
    CONCAT_WS(', ',
            customers.Customer_FName,
            customers.Customer_LName) AS 'Customer Name',
    order_details.Product_id,
    order_details.Quantity,
    products.Product_Name,
    suppliers.Suppliers_id,
    suppliers.Company_Name
FROM
    orders
        INNER JOIN
    customers ON orders.Customer_id = customers.Customer_id
        INNER JOIN
    order_details ON orders.Order_id = order_details.Order_id
        INNER JOIN
    products ON order_details.Product_id = products.Product_id
        INNER JOIN
    suppliers ON products.Suppliers_id = suppliers.Suppliers_id
    Where
    orders.Order_id = Order_ID AND
    orders.Customer_id = Cust_ID;

END
```

**Output:**

```
1 • call SP_Order_Details_Retrival(4518, 1372);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Order_id | Customer_id | Customer Name | Product_id | Quantity | Product_Name | Suppliers_id | Company_Name | Contact_Name | Supplier_EMail | Supplier_Phone |
|---|---|---|---|---|---|---|---|---|---|---|
| 4518 | 1372 | Carly. Snow | 20007 | 10 | Product 7 | 30810 | Google | Larry P | info@google.com | 857-763-0002 |
| 4518 | 1372 | Carly. Snow | 20008 | 12 | Product 8 | 30810 | Google | Larry P | info@google.com | 857-763-0002 |
| 4518 | 1372 | Carly. Snow | 20018 | 10 | Product 7 | 30830 | Apple | Tim Cook | info@apple.com | 857-763-0004 |
| 4518 | 1372 | Carly. Snow | 20025 | 12 | Product 7 | 30840 | HTC | John H | info@htc.com | 857-763-0005 |

Use: Can be used by customer care team to contact with supplier in any concern raise by the customer.

**2. SP_Track_Order :** Used to display shipment status of a an order.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SP_Track_Order`(IN Order_ID INT)
BEGIN
SELECT
    order_delivery.Order_id,
  --  order_delivery.Delivery_Status_id,
    delivery_status.Delivery_Status_Description,
    order_delivery.Date
FROM
    order_delivery,
    delivery_status
WHERE
    order_delivery.Delivery_Status_id = delivery_status.Delivery_Status_id
        AND order_delivery.Order_id = Order_ID;

END
```

**Output**:

```
1 • call SP_Track_Order(4500);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Order_id | Delivery_Status_Description | Date |
|---|---|---|
| 4500 | Preparing for shipment | 2017-12-11 18:23:25 |
| 4500 | Pakaging | 2017-12-11 18:23:30 |

Use: Can be used by customer care team or end user to track the shipping status of the product.

# 6. View

*"In a database, a view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. Views can join and simplify multiple tables into a single virtual table."*

Created Views:

**1. customer_order_prodcut_supplier_details:** used to display order of the customer along with quantity and supplier details.

```sql
CREATE VIEW `Customer_Order_Prodcut_Supplier_Details` AS
    SELECT
        orders.Order_id,
        orders.Customer_id,
        CONCAT_WS(', ',
                customers.Customer_FName,
                customers.Customer_LName) AS 'Customer Name',
        order_details.Product_id,
        order_details.Quantity,
        products.Product_Name,
        suppliers.Suppliers_id,
        suppliers.Company_Name
    FROM
        orders
            INNER JOIN
        customers ON orders.Customer_id = customers.Customer_id
            INNER JOIN
        order_details ON orders.Order_id = order_details.Order_id
            INNER JOIN
        products ON order_details.Product_id = products.Product_id
            INNER JOIN
        suppliers ON products.Suppliers_id = suppliers.Suppliers_id;
```

**Output:**

```sql
select * from customer_order_prodcut_supplier_details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Order_id | Customer_id | Customer Name | Product_id | Quantity | Product_Name | Suppliers_id | Company_Name |
|---|---|---|---|---|---|---|---|
| 4515 | 1324 | Larry. Schroeder | 20004 | 3 | Product 4 | 30800 | Facebook |
| 4522 | 1979 | Brooks. Lindsey | 20002 | 3 | Product 2 | 30800 | Facebook |
| 4518 | 1372 | Carly. Snow | 20007 | 10 | Product 7 | 30810 | Google |
| 4518 | 1372 | Carly. Snow | 20008 | 12 | Product 8 | 30810 | Google |
| 4522 | 1979 | Brooks. Lindsey | 20008 | 15 | Product 8 | 30810 | Google |
| 4518 | 1372 | Carly. Snow | 20018 | 10 | Product 7 | 30830 | Apple |
| 4518 | 1372 | Carly. Snow | 20025 | 12 | Product 7 | 30840 | HTC |

# 7. Analytics

1. Find name of the company in descending order with respect to total sales.

```
SELECT
   products.Suppliers_id,
   suppliers.Company_Name,
   SUM(order_details.Total) AS `Total Sales`
FROM
   order_details
      INNER JOIN
   products ON order_details.Product_id = products.Product_id
      INNER JOIN
   suppliers ON products.Suppliers_id = suppliers.Suppliers_id
GROUP BY products.Suppliers_id , suppliers.Company_Name
ORDER BY `Total Sales` DESC;
```

```
SELECT
    products.Suppliers_id,
    suppliers.Company_Name,
    SUM(order_details.Total) AS `Total Sales`
FROM
    order_details
        INNER JOIN
    products ON order_details.Product_id = products.Product_id
        INNER JOIN
    suppliers ON products.Suppliers_id = suppliers.Suppliers_id
GROUP BY products.Suppliers_id , suppliers.Company_Name
ORDER BY `Total Sales` DESC;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

| Suppliers_id | Company_Name | Total Sales |
|---|---|---|
| 30810 | Google | 60374.85 |
| 30840 | HTC | 44858.64 |
| 30830 | Apple | 13505.50 |
| 30800 | Facebook | 2598.30 |

2. Find top 10 product categories with respect to product sold.

```
SELECT
   products.Category_id,
   categories.Category_Name,
   SUM(order_details.Quantity) AS `Total Quantity Sold`
FROM
   order_details
      INNER JOIN
   products ON order_details.Product_id = products.Product_id
      INNER JOIN
   categories ON products.Category_id = categories.Category_id
GROUP BY products.Category_id , categories.Category_Name
ORDER BY `Total Quantity Sold` DESC
LIMIT 10;
```

```
SELECT
    products.Category_id,
    categories.Category_Name,
    SUM(order_details.Quantity) AS `Total Quantity Sold`
FROM
    order_details
        INNER JOIN
    products ON order_details.Product_id = products.Product_id
        INNER JOIN
    categories ON products.Category_id = categories.Category_id
GROUP BY products.Category_id , categories.Category_Name
ORDER BY `Total Quantity Sold` DESC
LIMIT 10;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

| Category_id | Category_Name | Total Quantity Sold |
|---|---|---|
| 10002 | Shopping Products | 52 |
| 10001 | Convenience Products | 10 |
| 10004 | Emergency Products | 6 |