Case Study Article on E-Commerce Database Design

TRENDIFY – Empowering Online Fashion Retail

Written by:
Lovepreet Singh

Table of Contents

Introduction

1.1 Purpose of the Database Design

The purpose of this database design is to build a comprehensive and efficient system for TRENDIFY, an innovative online fashion retailer. By structuring data effectively, the design aims to streamline key operations such as order processing, inventory management, payment handling, and customer engagement. This results in a smooth, reliable, and scalable solution that enhances the overall shopping experience.

1.2 Overview of TRENDIFY

TRENDIFY is a modern e-commerce platform focused on delivering high-quality, trendy fashion at competitive prices. With a strong commitment to fast shipping, customer satisfaction, and innovation, TRENDIFY leverages data-driven decision-making to stay ahead in a rapidly evolving market. The database serves as the backbone of these operations, ensuring that every transaction and interaction is recorded accurately and processed efficiently.

2. Mission and Objectives

2.1 Mission Statement

TRENDIFY's mission is to provide a seamless online shopping experience by offering stylish, high-quality apparel at great prices. The company is dedicated to innovation, rapid service, and outstanding customer support, ensuring that every customer enjoys a personalized and hassle-free shopping journey.

2.2 Business Objectives

To fulfill its mission, TRENDIFY's database design focuses on:

- **Efficient Order Management:** Capturing and tracking every order accurately to ensure prompt processing and delivery.
- **Robust Inventory Control:** Maintaining real-time inventory records to avoid stockouts and overstocking while facilitating timely reordering.
- **Secure Payment Processing:** Ensuring that payment transactions are handled securely and reliably to build customer trust.
- **Enhanced Customer Engagement:** Storing detailed customer information to enable personalized services, targeted promotions, and effective communication.
- **Streamlined Supplier and Logistics Coordination:** Tracking supplier details and shipment statuses to ensure a smooth and responsive supply chain.

---

3. Database Design Overview

3.1 Identified Tables

Based on TRENDIFY's operational needs, the database design includes the following core tables:

- **Customers:** Contains personal and contact information for shoppers.
- **Products:** Holds details about each product, including descriptions, pricing, and stock levels.
- **Orders:** Records customer orders with details such as order date, items purchased, and total amount.
- **Payments:** Stores payment transaction information to verify and process payments securely.
- **Shipments:** Tracks delivery information and shipment statuses.
- **Employees:** Contains details about staff responsible for order processing and customer service.
- **Suppliers:** Manages data on product suppliers and sourcing information.
- **Offers:** Maintains information on discounts, promotions, and special deals.

3.2 Focus Areas

The design is primarily focused on:

- **Order Management:** Ensuring that every order is recorded and processed efficiently.
- **Inventory Tracking:** Keeping real-time data on product availability to manage stock effectively.
- **Payment Handling:** Safeguarding and processing payment information with high security.
- **Customer Data Management:** Utilizing customer information to drive personalized marketing and improved service.
- **Operational Reporting:** Creating dynamic views for invoices, order details, and inventory to aid in decision-making.

4. Tables and Their Attributes

4.1 Customers Table

- **Purpose:** To store comprehensive customer data for efficient service and personalized engagement.
- **Key Attributes:** Customer_ID, First_Name, Last_Name, Email, Phone, Address

4.2 Products Table

- **Purpose:** To manage detailed information about each product, including stock levels.
- **Key Attributes:** Product_ID, Product_Name, Description, Price, Quantity_in_Stock

4.3 Orders Table

- **Purpose:** To capture every customer order and the associated details for seamless processing.
- **Key Attributes:** Order_ID, Customer_ID, Order_Date, Total_Amount

4.4 Payments Table

- **Purpose:** To record secure payment transactions and related financial data.
- **Key Attributes:** Payment_ID, Order_ID, Payment_Method, Payment_Date, Total_Amount

4.5 Shipments Table

- **Purpose:** To monitor the delivery process and ensure timely order fulfillment.
- **Key Attributes:** Shipment_ID, Order_ID, Shipment_Date, Delivery_Status

4.6 Employees Table

- **Purpose:** To store employee information for operational management and customer support.
- **Key Attributes:** Employee_ID, First_Name, Last_Name, Role, Contact_Info

4.7 Suppliers Table

- **Purpose:** To track supplier information, ensuring efficient product sourcing and replenishment.
- **Key Attributes:** Supplier_ID, Supplier_Name, Contact_Details, Product_List

4.8 Offers Table

- **Purpose:** To manage promotional campaigns and discounts available to customers.
- **Key Attributes:** Offer_ID, Description, Discount_Percentage, Valid_From, Valid_To

---

5.LIST OF TABLES

1. Customers
2. Employees
3. Suppliers
4. Products
5. Orders
6. Payment Methods
7. Shipments
8. Offers

5.2 LIST OF TABLES AND ATTRIBUTES

Customer Table:

Customer_ID	Primary Key)
First_Name	
Last_Name	
Email	
Phone	
Address (House Number, Street, City, Province, Zip Code)	

Employee Table:

Employee_ID	(Primary Key)
Name	
Role	
Skills	
Availability	
Hire Date	
Resign Date	
Salary	

Product Table:

Product_ID	(Primary Key)
Product_Name	
Description	
Quantity_In_Stock	
Price	
Supplier_ID	(Foreign Key)

Order Table:

Order_ID	(Primary Key)
Customer_ID	(Foreign Key)
Order_Date	
QUANTITY	
Product_ID	(Foreign Key)
Employee_ID	(Foreign Key)
Shipment_ID	(Foreign Key)
Offer_ID	(Foreign Key)

Shipment Table:

Shipment_ID	(Primary Key)
Shipment_Date	
Tracking_Number	
Status (In Transit, Delivered, Returned)	

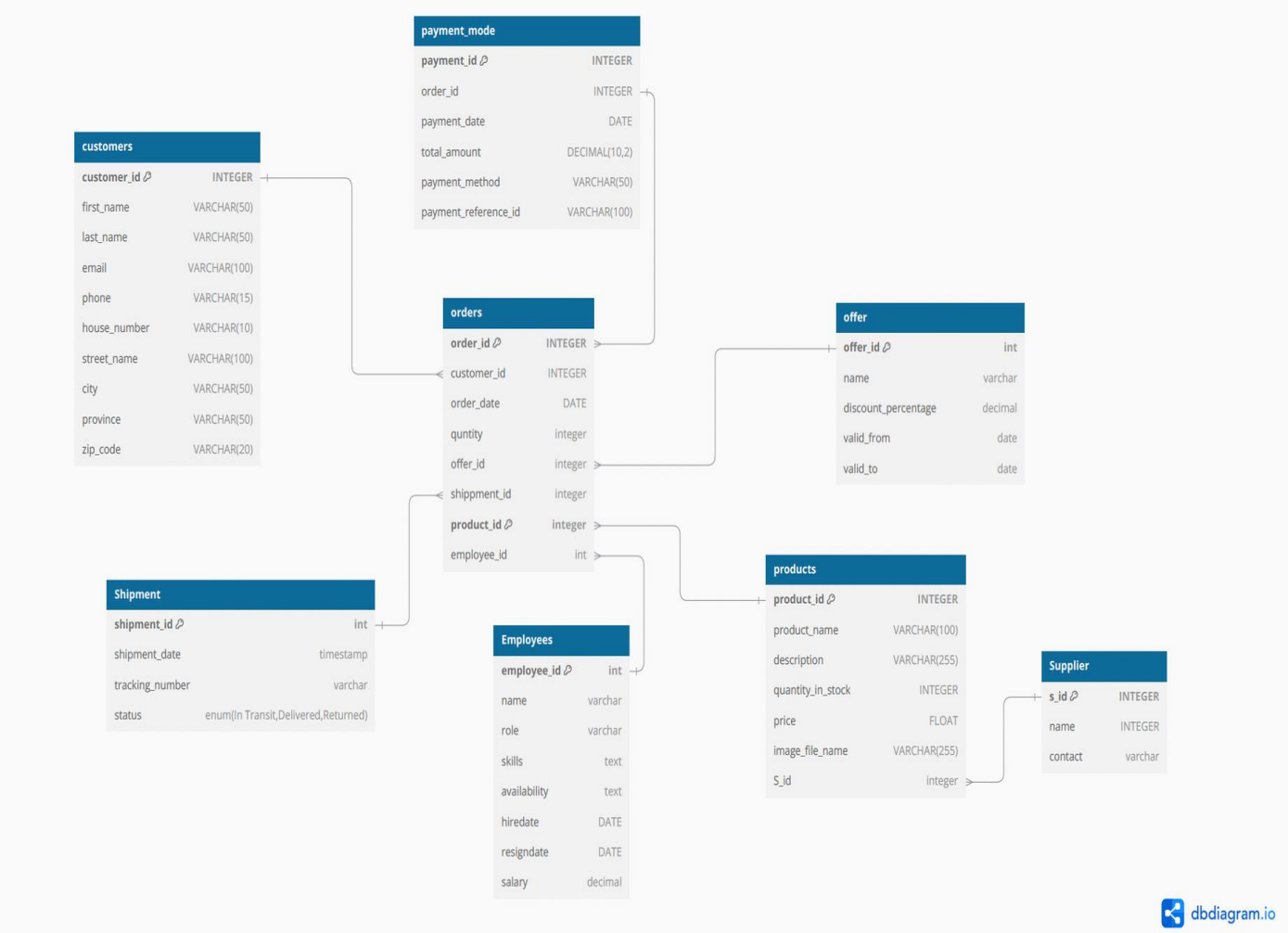
Payment Table:

Payment_ID	(Primary Key)
Order_ID	(Foreign Key)
Payment_Date	
Total_Amount	
Payment_Method	
Payment_Reference_ID	

Offer Table:

Offer_ID	(Primary Key)
Name	
Discount_Percentage	
Valid_From	
Valid_To	

6.1Entity Relationship Diagram (ERD)



6.2 Overview of the ERD

The ERD visually represents the database structure, outlining how each table is interconnected. Key entities such as Customers, Orders, and Products are mapped along with their relationships. This visual tool is critical for ensuring that data flows correctly throughout the system and for simplifying future modifications or expansions.

6.3 Relationships Between Tables

The TRENDIFY database establishes several key relationships to maintain data integrity and streamline operations:

- **Customers to Orders:** A one-to-many relationship where a single customer can place multiple orders.
- **Orders to Payments:** A one-to-one relationship that ensures each order is linked to a specific payment record.
- **Orders to Products:** Managed through a junction (or order details) table, this many-to-many relationship allows an order to include multiple products while a product may appear in multiple orders.
- **Products to Suppliers:** Each product is associated with a supplier, ensuring accurate inventory sourcing.

Employees to Orders: A one-to-many relationship where each employee can manage multiple orders.

7.1 INVOICE VIEWS

SCENARIO

To know the total amount invoiced per customer, including payment details, unit price, and total price calculation.

TABLES USED

Orders (orders)Customers (customers)Products (products)Payment Mode (payment_mode)

FIELDS USED

order_id,customer_name,email,order_date,product_name, quantity,payment_method,payment_date,total_amount.

PURPOSE

To gather information for tracking invoices per customer by including customer details, order items, and payment transactions

```sql
CREATE VIEW InvoiceView AS
SELECT
    o.order_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    c.email,
    o.order_date,
    p.product_name,
    o.quantity,
    pm.payment_method,
    pm.payment_date,
    pm.total_amount
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
JOIN products p ON o.product_id = p.product_id
JOIN payment_mode pm ON o.order_id = pm.order_id;

-- Step 5: Verify Data
SELECT * FROM customers;
SELECT * FROM products;
SELECT * FROM orders;
SELECT * FROM payment_mode;

-- Step 6: Query the Invoice View
SELECT * FROM InvoiceView;
```

	order_id	customer_name	email	order_date	product_name	quantity	payment_method	payment_date	total_amount
1	1	John Doe	john@example.com	2024-01-29	Red Sneakers	2	Credit Card	2024-01-29	99.98
2	2	Jane Smith	jane@example.com	2024-01-30	White T-Shirt	1	PayPal	2024-01-30	19.99

7.2 ORDER DETAIL'S VIEW

Scenario

To know the detailed order information, including customer name, product details, unit price, total price, and order date.

Purpose

To gather detailed order information, providing insights into customer purchases, pricing, and transaction history.

```
108  CREATE VIEW OrderDetailsView AS
109  SELECT
110      o.order_id,
111      CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
112      p.product_name,
113      o.quantity,
114      p.price AS unit_price,
115      (o.quantity * p.price) AS total_price,
116      o.order_date,
117      s.tracking_id,
118      s.status AS shipment_status
119  FROM orders o
120  JOIN customers c ON o.customer_id = c.customer_id
121  JOIN products p ON o.product_id = p.product_id
122  LEFT JOIN shipments s ON o.order_id = s.order_id;
123
124  SELECT * FROM OrderDetailsView;
125
```

Results grid

	product_id	product_name	quantity_in_stock	supplier_name	price
1	1	Red Sneakers	100	Nike	49.99
2	2	White T-Shirt	50	Adidas	19.99

7.3 INVENTORY VIEW

SCENARIO:

We want to track product inventory, including available stock, supplier details, and pricing information.

PURPOSE:

To gather real-time inventory data, for accurate tracking of stock levels, supplier sources, and pricing

```
--Create Inventory View
CREATE VIEW InventoryView AS
SELECT
    p.product_id,
    p.product_name,
    COALESCE(p.quantity_in_stock, 0) AS quantity_in_stock, -- Prevent NULL values
    COALESCE(s.name, 'Unknown Supplier') AS supplier_name, -- Default supplier name if missing
    p.price
FROM products p
LEFT JOIN supplier s ON p.s_id = s.s_id;

--
SHOW FULL TABLES IN shein WHERE TABLE_TYPE LIKE 'VIEW';
DESC products;

--Show Inventory View
SELECT * FROM InventoryView;
```

	order_id	customer_name	email	order_date	product_name	quantity	payment_method	payment_date	total_amount
1	1	John Doe	john@example.com	2024-01-29	Red Sneakers	2	Credit Card	2024-01-29	99.98
2	2	Jane Smith	jane@example.com	2024-01-30	White T-Shirt	1	PayPal	2024-01-30	19.99

7. Conclusion

The TRENDIFY e-commerce database design offers a robust, scalable solution tailored to the unique needs of an online fashion retailer. By organizing data across essential operational areas—order processing, inventory management, payment handling, and customer engagement—the system ensures efficient, error-free operations and improved customer satisfaction. With clear relationships between tables and dynamic reporting views, the database not only meets current demands but also sets a solid foundation for future innovations, such as automated inventory alerts and advanced analytics. Ultimately, this design empowers TRENDIFY to remain agile and competitive in a fast-paced digital marketplace.