

# Database Schema Documentation

## 1. Entity-Relationship Description

### ENTITY: Customers

**Purpose:** Stores customer information.

**Attributes:** - `customer_id` (INT, Primary Key): Unique identifier for each customer - `first_name` (VARCHAR): Customer's first name - `last_name` (VARCHAR): Customer's last name - `email` (VARCHAR, Unique): Customer's email address - `phone` (VARCHAR): Customer's contact number - `address` (TEXT): Customer's residential address - `created_at` (DATETIME): Date and time when the customer account was created

**Relationships:** - One customer can place **many orders** (1:M relationship with Orders table)

---

### ENTITY: Orders

**Purpose:** Stores order details placed by customers.

**Attributes:** - `order_id` (INT, Primary Key): Unique identifier for each order - `customer_id` (INT, Foreign Key): References `customers.customer_id` - `order_date` (DATETIME): Date and time when the order was placed - `order_status` (VARCHAR): Current status of the order (e.g., Pending, Shipped, Delivered) - `total_amount` (DECIMAL): Total cost of the order

**Relationships:** - Each order is placed by **one customer** (M:1 relationship with Customers table) - One order can contain **many order items** (1:M relationship with Order\_Items table)

---

### ENTITY: Products

**Purpose:** Stores product information available for purchase.

**Attributes:** - `product_id` (INT, Primary Key): Unique identifier for each product - `product_name` (VARCHAR): Name of the product - `description` (TEXT): Product description - `price` (DECIMAL): Price of the product - `stock_quantity` (INT): Available quantity in stock

**Relationships:** - One product can appear in **many order items** (1:M relationship with Order\_Items table)

---

### ENTITY: Order\_Items

**Purpose:** Stores details of products included in an order.

**Attributes:** - `order_item_id` (INT, Primary Key): Unique identifier for each order item - `order_id` (INT, Foreign Key): References `orders.order_id` - `product_id` (INT, Foreign Key): References `products.product_id` - `quantity` (INT): Number of units ordered - `price` (DECIMAL): Price of the product at the time of order

**Relationships:** - Each order item belongs to **one order** (M:1 relationship with Orders table) - Each order item references **one product** (M:1 relationship with Products table)

---

## 2. Summary of Relationships

- Customers ↔ Orders: One-to-Many (1:M)
  - Orders ↔ Order\_Items: One-to-Many (1:M)
  - Products ↔ Order\_Items: One-to-Many (1:M)
- 

*This schema supports an online shopping system by efficiently managing customers, orders, and products.*

---

## 2. Normalization Explanation (3NF)

The given database design follows **Third Normal Form (3NF)**, ensuring minimal redundancy and improved data integrity. First Normal Form (1NF) is achieved because all tables contain atomic values, and there are no repeating groups or multivalued attributes. Each field stores only a single value, such as one email per customer or one product per order item row.

Second Normal Form (2NF) is satisfied as all non-key attributes are fully functionally dependent on the entire primary key. For example, in the `Order_Items` table, attributes like `quantity` and `price` depend on `order_item_id` and not partially on `order_id` or `product_id`. This eliminates partial dependency issues.

Third Normal Form (3NF) is achieved because there are no transitive dependencies. In the `Customers` table, attributes such as `first_name`, `email`, and `address` depend only on `customer_id`. Similarly, in the `Orders` table, `order_status` and `total_amount` depend only on `order_id`, not indirectly through other attributes.

**Functional Dependencies:** - `customer_id → first_name, last_name, email, phone, address` - `order_id → customer_id, order_date, order_status, total_amount` - `product_id → product_name, description, price, stock_quantity` - `order_item_id → order_id, product_id, quantity, price`

---

This design avoids **update anomalies** by ensuring that data changes (e.g., customer email updates) are made in only one place. **Insert anomalies** are prevented because new customers or products can be added independently. **Delete anomalies** are avoided since deleting an order does not remove customer or product information.

---

### 3. Sample Data Representation

**Customers Table**

customer_id	first_name	last_name	email	phone
1	Rahul	Sharma	rahul@gmail.com	9876543210
2	Sneha	Patil	sneha@gmail.com	9123456780

**Orders Table**

order_id	customer_id	order_date	order_status	total_amount
101	1	2025-01-10	Delivered	2500.00
102	2	2025-01-12	Shipped	1800.00

**Products Table**

product_id	product_name	price	stock_quantity
201	Headphones	1500.00	20
202	Keyboard	1200.00	15

**Order\_Items Table**

order_item_id	order_id	product_id	quantity	price
1	101	201	1	1500.00
2	102	202	1	1200.00

## 2. Normalization Explanation (3NF)

The database design follows **Third Normal Form (3NF)** to ensure data integrity, reduce redundancy, and avoid anomalies.

**First Normal Form (1NF):** All tables have atomic attributes, meaning each field contains indivisible values. There are no repeating groups or multi-valued attributes. For example, product details and customer details are stored separately, and order items are stored in a dedicated table.

**Second Normal Form (2NF):** All non-key attributes are fully functionally dependent on the entire primary key. Composite relationships are resolved using separate tables such as `Order_Items`, ensuring that attributes like `quantity` and `price` depend on the whole key and not partially on `order_id` or `product_id` alone.

**Third Normal Form (3NF):** There are no transitive dependencies. Non-key attributes depend only on the primary key. For instance, customer details are stored only in the `Customers` table and not

repeated in the **Orders** table. Similarly, product price and description are stored only in the **Products** table.

**Functional Dependencies:** -  $customer\_id \rightarrow first\_name, last\_name, email, phone, address$  -  $order\_id \rightarrow order\_date, order\_status, total\_amount, customer\_id$  -  $product\_id \rightarrow product\_name, description, price, stock\_quantity$  -  $order\_item\_id \rightarrow order\_id, product\_id, quantity, price$

**Avoidance of Anomalies:** - **Update anomalies** are avoided because data is stored in only one place (e.g., product price). - **Insert anomalies** are prevented as new customers or products can be added without placing an order. - **Delete anomalies** are avoided since deleting an order does not remove customer or product information.

### 3. Sample Data Representation

#### Customers Table

customer_id	first_name	last_name	email	phone	address
1	Rahul	Sharma	rahul@gmail.com	9876543210	Pune
2	Ananya	Verma	ananya@gmail.com	9123456780	Mumbai

#### Orders Table

order_id	customer_id	order_date	order_status	total_amount
101	1	2025-01-05	Delivered	2499.00
102	2	2025-01-06	Pending	1599.00

#### Products Table

product_id	product_name	description	price	stock_quantity
201	Wireless Mouse	Optical mouse	799.00	50
202	Keyboard	Mechanical keyboard	1699.00	30

#### Order\_Items Table

order_item_id	order_id	product_id	quantity	price
1	101	201	1	799.00
2	102	202	1	1699.00