**Case Study: Amazon Database Management**

**Author- Mitul Makwana**

---

**Table of Contents**

---

## 1. Introduction

Amazon's databases manage a vast range of information, from customer profiles and product listings to orders and shipping details. Efficient database design allows Amazon to provide personalized recommendations, streamline inventory, and enhance customer satisfaction while ensuring data security.

---

## 2. Mission Statement

To deliver a seamless and personalized shopping experience, enabling customers to find and purchase products effortlessly.

---

## 3. Objectives

- **Inventory Management**: Ensuring stock availability and optimizing restocks.

- **Operational Efficiency**: Streamlining orders, payments, and shipping.

- **Customer Experience**: Enhancing personalization and improving product recommendations.
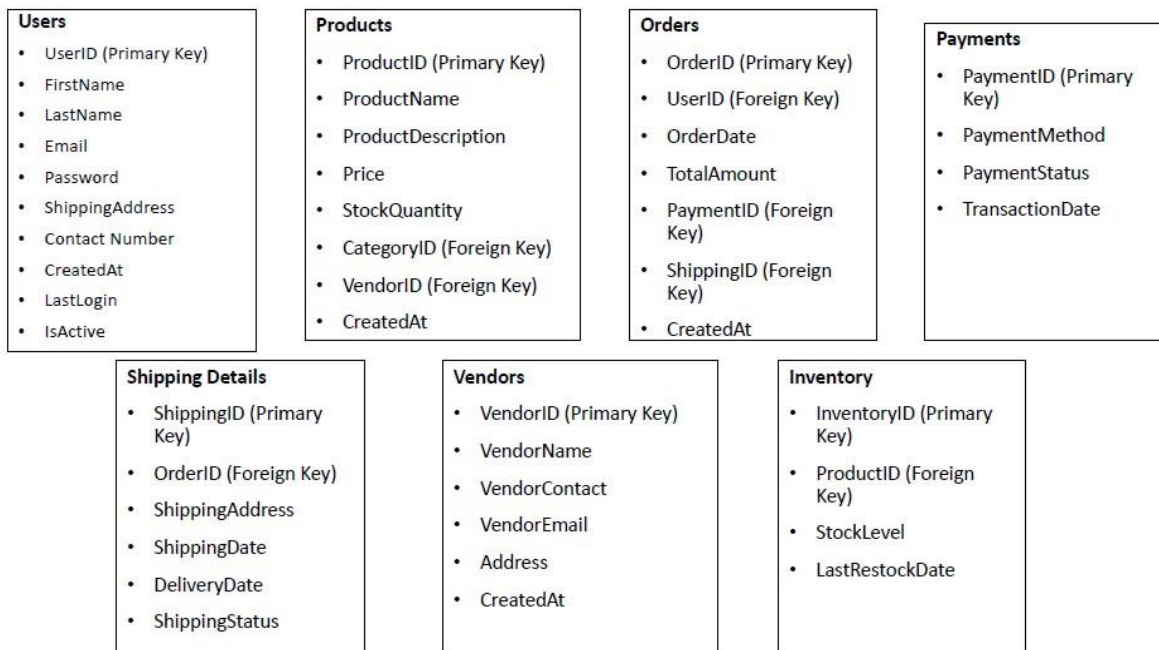
---

## 4. Database Structure

**Tables**

- **Users**: Stores customer data such as UserID, name, contact details, etc.

- **Products**: Manages product information including ProductID, price, stock, and vendor details.
- **Orders**: Tracks each order with related data like OrderID, UserID, and payment status.

- **Payments**: Stores transaction details, including payment methods and status.

- **Shipping Details**: Manages shipping status and addresses.

- **Vendors**: Contains vendor-specific information such as VendorID and contact details.

- **Inventory**: Tracks stock levels and restocking details.

- **Categories**: Organizes products under different categories.

### 4.1 Core Components

# Fields in Tables

**Users**
- UserID (Primary Key)
- FirstName
- LastName
- Email
- Password
- ShippingAddress
- Contact Number
- CreatedAt
- LastLogin
- IsActive

**Products**
- ProductID (Primary Key)
- ProductName
- ProductDescription
- Price
- StockQuantity
- CategoryID (Foreign Key)
- VendorID (Foreign Key)
- CreatedAt

**Orders**
- OrderID (Primary Key)
- UserID (Foreign Key)
- OrderDate
- TotalAmount
- PaymentID (Foreign Key)
- ShippingID (Foreign Key)
- CreatedAt

**Payments**
- PaymentID (Primary Key)
- PaymentMethod
- PaymentStatus
- TransactionDate

**Shipping Details**
- ShippingID (Primary Key)
- OrderID (Foreign Key)
- ShippingAddress
- ShippingDate
- DeliveryDate
- ShippingStatus

**Vendors**
- VendorID (Primary Key)
- VendorName
- VendorContact
- VendorEmail
- Address
- CreatedAt

**Inventory**
- InventoryID (Primary Key)
- ProductID (Foreign Key)
- StockLevel
- LastRestockDate

### 4.2 Table Relationships

- **Users to Orders**: One-to-Many

- **Products to Vendors**: Many-to-One

- **Orders to Payments**: One-to-One

- **Orders to Shipping**: One-to-One

**4.3 ENTITY RELATIONSHIP DIAGRAM(ERD)**

In my project, I worked with cloud platforms like Azure SQL to set up databases for managing e-commerce data. I also used relational databases like PostgreSQL and MySQL to handle large datasets more efficiently.
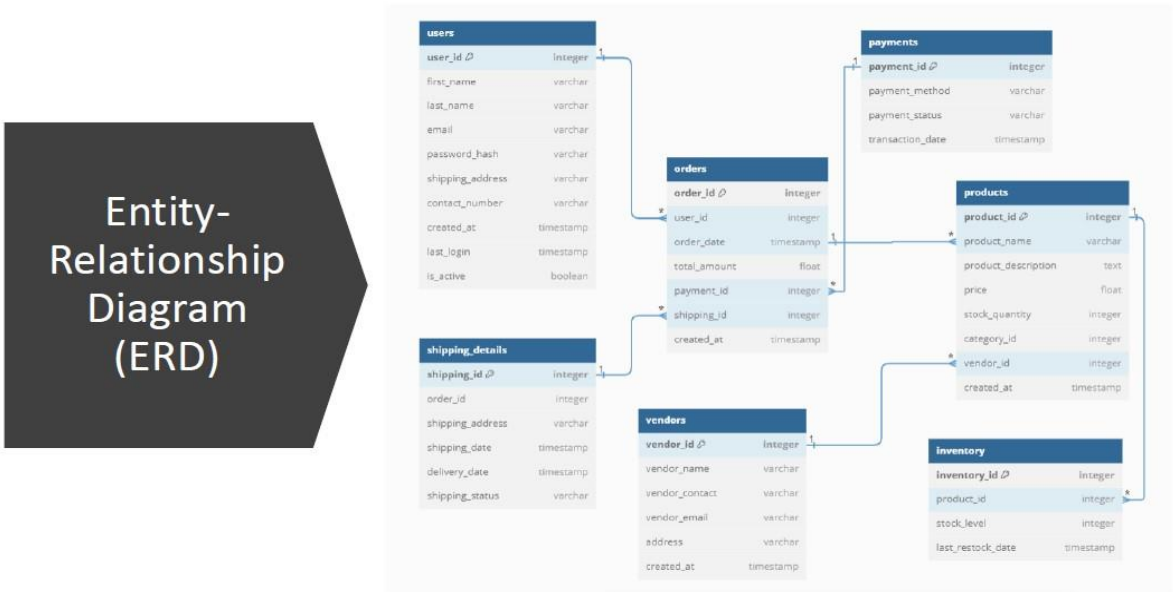


# 5. Conclusion

Amazon's database design supports millions of transactions daily, ensuring smooth order management and data security. The system is optimized for scalability, data integrity, and operational efficiency.

### 6. Appendix: SQL Query Components

| Component | Description |
| --- | --- |
| Query Purpose | Retrieve detailed information about users' orders, including payment and shipping details. |
| Selected Fields | - Users.UserID<br>- Users.FirstName<br>- Users.LastName<br>- Orders.OrderID<br>- Orders.OrderDate<br>- Orders.TotalAmount<br>- Payments.PaymentMethod<br>- Payments.PaymentStatus<br>- ShippingDetails.ShippingAddress |

| | |
|---|---|
| | - ShippingDetails.ShippingDate |
| | - ShippingDetails.DeliveryDate |
| | - ShippingDetails.ShippingStatus |
| **Main Tables** | **Users**<br>**Orders**<br>**Payments**<br>- <br>- <br>- <br>- <br>- **ShippingDetails** |
| **Relationships** | - Users to Orders: One-to-Many (One user can have multiple orders) - Orders to Payments: One-to-One (Each order has one payment) - Orders to ShippingDetails: One-to-One (Each order has one shipping detail) |
| **Join Conditions** | - Orders.UserID = Users.UserID<br>- Orders.PaymentID = Payments.PaymentID<br>- Orders.OrderID = ShippingDetails.OrderID |
| **SQL Clauses** | - **SELECT**: Specifies the columns to be retrieved.<br>- **FROM**: Indicates the primary table (Orders) to retrieve data from. - **JOIN**: Combines rows from two or more tables based on related columns. |
| **Output Expected** | A comprehensive list of orders, including user information, payment details, and shipping statuses. |
| **Potential Use Cases** | - Analyze user purchasing behavior<br>- Monitor payment and shipping statuses<br>- Generate reports for customer service inquiries |

## 7. Appendix Table

| Section | Content |
|---|---|
| **Tables Used** | Users, Products, Orders, Vendors, Order Items, Shipping Details, Reviews |
| **Key Relationships** | Users ↔ Orders (One-to-Many), Orders ↔ Order Items (One-to-Many), Products ↔ Vendors (Many-to-One), Orders ↔ Shipping Details (Oneto-One), Products ↔ Reviews (One-to-Many) |

## Database Platforms

AWS RDS, Azure SQL, PostgreSQL, MySQL

**Key Objectives**    Ease of Use, Efficiency, Security, Scalability, Personalization

**Core Components** Users, Products, Orders, Payments, Shipping Details

**ERD Tools Used**    Azure SQL, PostgreSQL, MySQL
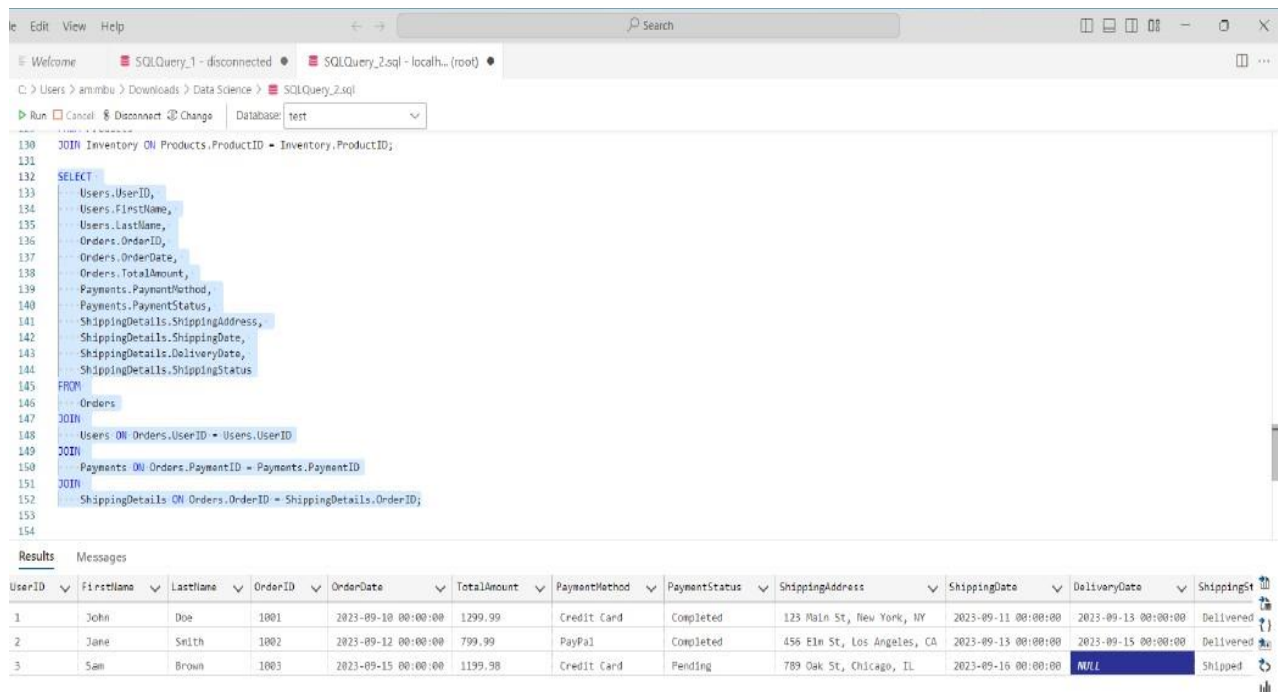
**Security Measures**  Data encryption, Secure payment processing, Privacy Protocols

**Personalization Methods**    Recommendations based on browsing history, purchase history, and user preferences

**Scalability**    Designed for millions of users and products, with the ability to grow alongside the platform's needs

**Technologies Involved**    Cloud Platforms (AWS, Azure), Relational Databases (PostgreSQL, MySQL), Real-time Data Processing

**Conclusion Summary**    A scalable, secure, and efficient database designed to handle Amazon's e-commerce platform growth and needs.



**Contact:**
Email- mitul22111997@gmail.com

LinkedIn