A GUIDE TO DATABASE DESIGN

Case study of FitFlex Stores

By: Omololu A. Fashokun
January,2025.

Table of Content

About the Company

Fitflex is a dynamic e-commerce platform dedicated to empowering fitness enthusiasts, athletes, and health-conscious individuals by providing high-quality fitness products and services. It positions itself as a one-stop destination for all fitness needs, combining convenience, innovation, and customer-centric features to enhance the shopping experience.

Company Mission Statement.

Our mission is to empower every sport enthusiast by offering a wide range of high-performance and durable sport gear and equipment designed to meet the demands of any sport.

Project Mission Statement.

Designing a secure, scalable, and efficient database for FitGear to optimize operations, enhance decision-making, and drive business growth in the fitness e-commerce space,

Objectives.

I. Design and implement a normalized database schema.
II. Generate reports and dashboards to support business decisions.
III. Ensure reliable, structured, and secure storage of customer, product, and transaction data.

What is database?

A database is an organized collection of data used for the purpose of modeling some type of organization or organizational process. It really doesn't matter whether you're using spreadsheets or a database application program on the computer to collect and store the data. As long as you're gathering data in some organized manner for a specific purpose, you've got a database.

What is a good database design?

For a database design to be said to be a good design, Efficiency, Scalability, and ease of retrieval of information/ data amongst others are conditions that must be met. Integrity of data, constraints, relationships between tables, conforming to normalization rules A well-designed database is also scalable, able to handle growing data volumes without performance lapses. It aids prompt and seamless query processing and retrieval, optimizing both storage and access. Security about proper access control and encryption to protect sensitive information. Also, good design provides for easy updates and modifications to accommodate business dynamic needs

Therefore, a good database design is one that:

I. Maintain data integrity at all levels (field, table, and relationship levels) to ensure accuracy and completeness of information.
II. Consistently aligns with relevant business rules, providing valid and meaningful information.
III. Divide your information into subject-based tables to reduce redundant data.

IV. Scalability - design the database structure to facilitate future modifications and expansions in response to evolving business information requirements

Common Terminologies

I. Table: A table is the chief structure of a database, representing specific subjects, objects, or events. They help organize data in the database.
II. Field: A field in a database is a single piece of data or a specific attribute within a table. It stores actual data.
III. Record: A record (row) in a database is a complete set of related fields representing one item or entity.
IV. Keys: Keys in a database are attributes or combinations of attributes used to identify and establish relationships between tables uniquely.
V. Data: The value you store in a table. Data itself is meaningless.
VI. Information: This is the organized and processed data stored in tables. Information is meaningful.
VII. Primary Key: A primary key in a database is a unique identifier for each record in a table.
VIII. Foreign Key: A foreign key in a database is a field that links to the primary key of another table to establish a relationship between the two tables.
IX. Relationships: Relationships in a database connect tables through keys to organize and manage related data.
X. Nulls: Nulls in a database represent missing or undefined values in a field.
XI. Views: Views in a database are virtual tables created by querying and combining data from other tables.
XII. Data Integrity: Data integrity in a database refers to the accuracy and consistency of stored data.

XIII. ERD (Entity Relationship Diagram): An ERD (Entity Relationship Diagram) in a database is a visual representation of the relationships between entities (tables) in a database.

XIV. Structured Query Language (SQL): Structured Query Language (SQL) is a programming language used to manage and manipulate relational databases.

PREDEFINED TABLES AND FIELD LIST

Preliminary Table List

- Product
- Customer
- Employee
- Sales Order
- Product Category
- Order Status
- Payment Mode
- Store
- Inventory.
- Supplier
- Shipment
- Return & Refund
- Loyalty Program

Predefined Tables

- Product
- Customer
- Employee
- Sales Order
- Product Category
- Order Status
- Payment Mode
- Store

Preliminary Field List

- OrderID
- CustomerID
- CustomerFirstName
- CustomerLastName
- EmployeeID
- EmployeeFirstName
- EmployeeLastName
- StoreID
- StoreName
- Order_Date
- OrderTime
- TotalAmount
- DiscountAmount
- OrderStatus
- PaymentMode
- TrackingNumber
- Employee Job_Role
- HireDate
- Salary
- Department
- ManagerID
- City
- Province/State
- PostalCode/Zip
- DateOfBirth
- MembershipType
- ProductID
- Product_Category
- Stock Quantity
- ProductName
- Description
- Price
- QuantityInStock
- CategoryID
- CategoryName
- Brand
- SupplierID
- CategoryID
- CategoryName
- ParentCategoryID
- StoreID
- StoreName
- Store_location

Predefined Field List

- ProductID (PK)
- Product_Name
- CategoryID
- Product_Category
- Price
- Stock Quantity
- CustomerID
- First_Name
- Last_Name
- E-mail
- Phone
- Address

'Same for Employee and Customer Entity Tables'

- Employee Job_Role
- StoreID
- OrderID
- Order_Date
- StatusID
- Order_Status
- PaymentID
- Payment_Mode

DATA DICTIONARY

CUSTOMER TABLE

Field Name	Data Type	Description
CustomerID	INT (PK)	Unique identifier for each customer.
First_Name	VARCHAR	First name of the customer.
Last_Name	VARCHAR	Last name of the customer.
E-mail	VARCHAR	Email address of the customer.
Address	VARCHAR	Residential address of the customer.
Contact No.	INT	Contact number of the customer.

PRODUCT TABLE

Field Name	Data Type	Description
ProductID	INT (PK)	Unique identifier for each product.
Product_Name	VARCHAR	Name of the product.
Price	DECIMAL	Price of the product.
CategoryID	INT (FK)	Unique Identifier for each category.

PRODUCT CATEGORY TABLE

Field Name	Data Type	Description
CategoryID	INT (PK)	Unique identifier for each category
Category_Name	VARCHAR	Associated Category name for each product

PAYMENT MODE TABLE

Field Name	Data Type	Description
PaymentID	INT (PK)	Unique identifier for each payment.
Payment_Type	VARCHAR	Mode of Payment associated.

ORDER STATUS TABLE

Field Name	Data Type	Description
StatusID	INT (PK)	Unique identifier for Order Status
Order_Status	VARCHAR	Status of the Order

STORE TABLE

Field Name	Data Type	Description
StoreID	INT (PK)	Unique identifier for each store.
Store_Name	VARCHAR	Name of the store.
Address	VARCHAR	Address of the Store
Contact_No	INT	Contact number of the store.

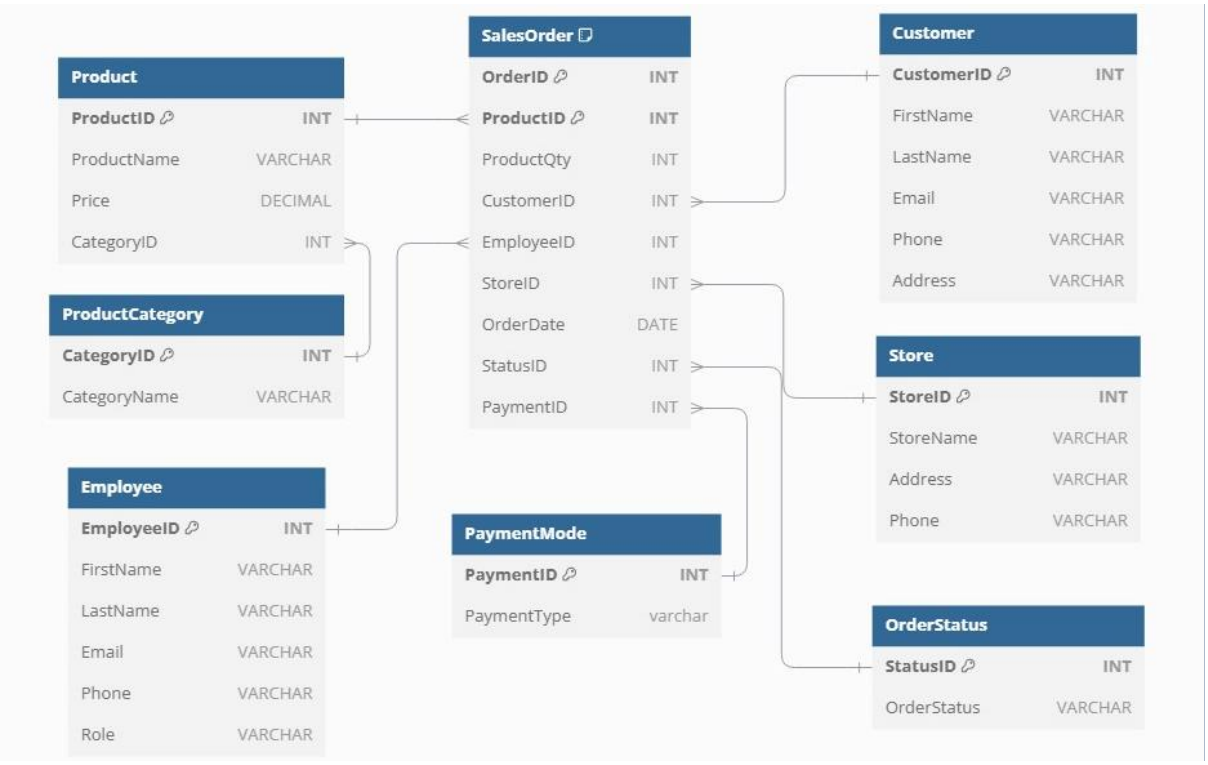
EMPLOYEE TABLE

Field Name	Data Type	Description
EmployeeID	INT (PK)	Unique identifier for each employee.
First_Name	VARCHAR	First name of the employee.
Last_Name	VARCHAR	Last name of the employee.
E-mail	VARCHAR	Email address of the employee.
Job_Role	VARCHAR	Employee's job role (e.g., Manager…)
Contact No.	INT	Contact number of the employee.

SALES ORDER TABLE

Field Name	Data Type	Description
OrderID	INT (CPK)	Unique identifier for each employee.
CustomerID	INT (FK)	Associated CustomerID
EmployeeID	INT (FK)	Associated EmployeeID
StoreID	INT (FK)	Associated StoreID
Order_Date	DATE	Date associated with the sales order.
ProductID	INT (CPK)	Associated ProductID
Product Quantity	INT	Quantity associated with the sale.
StatusID	INT (FK)	Associated Order StatusID
PaymentID	INT (FK)	Associated PaymentID

ENTITY RELATIONSHIP DIAGRAM (ERD)



RELATIONSHIPS AND DATABASE STRUCTURE

One-to-Many Relationships:

Customers → SalesOrders (One customer can place multiple orders)

SalesOrders → Products (One order can have multiple products)

SalesOrders → Store (One order is placed at a single store, but a store can process multiple orders)

ProductCategory → (Products (One category can have multiple products)

PaymentMode → (SalesOrders (One payment method can be used in multiple orders)

Store → (SalesOrders (One store can have multiple orders)

OrderStatus →(SalesOrders (One order status can be assigned to multiple orders)

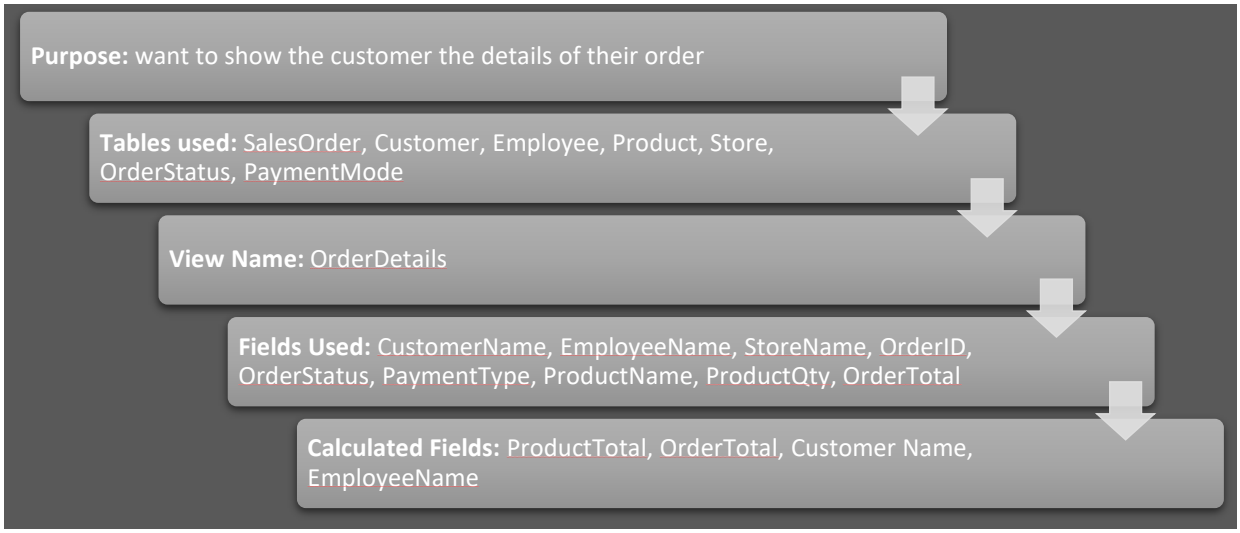
2. Many-to-Many Relationship & Solution:

Products & SalesOrders have a many-to-many relationship (one order can contain multiple products, and a product can appear in multiple orders).

To handle this, we introduce a **SalesOrderProducts** linking table between **SalesOrders** and **Products**.

VIEWS AND QUERIES

VIEW-ORDER DETAILS/CUSTOMER RECEIPT



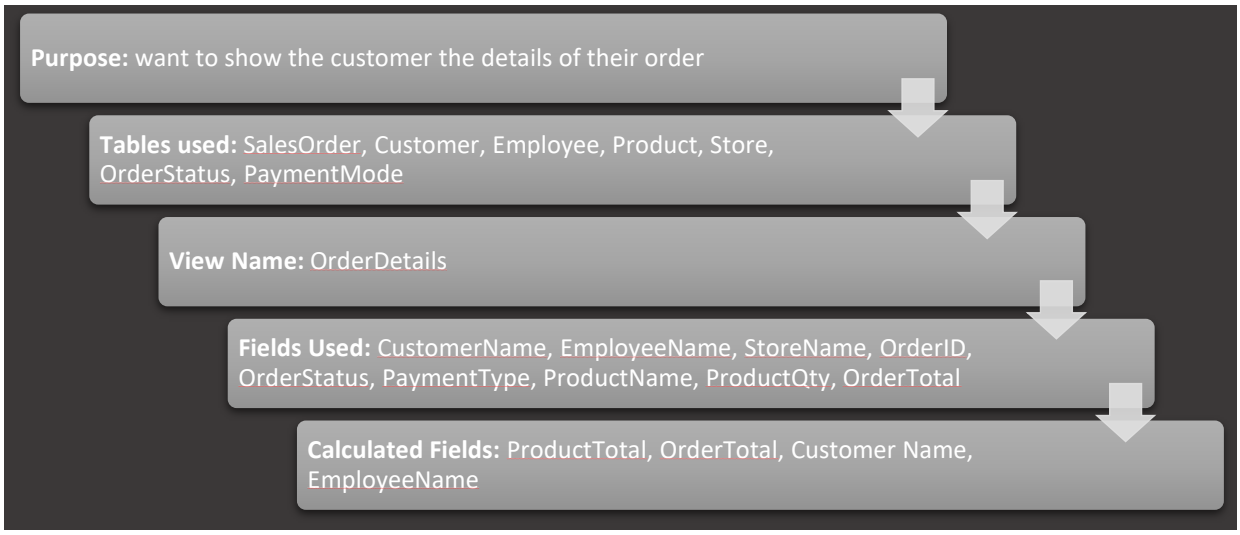
QUERIES

```
352  CREATE VIEW OrderDetails AS
353  SELECT
354      so.OrderID,
355      CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
356      CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,
357      s.StoreName,
358      os.OrderStatus,
359      pm.PaymentType,
360      p.ProductName,
361      so.ProductQty,
362      (p.Price * so.ProductQty) AS ProductTotal,
363      (SELECT SUM(p2.Price * so2.ProductQty)
364       FROM SalesOrder so2
365       JOIN Product p2 ON so2.ProductID = p2.ProductID
366       WHERE so2.OrderID = so.OrderID) AS OrderTotal
367  FROM
368      SalesOrder so
369  JOIN
370      Customer c ON so.CustomerID = c.CustomerID
371  JOIN
372      Employee e ON so.EmployeeID = e.EmployeeID
373  JOIN
374      Store s ON so.StoreID = s.StoreID
375  JOIN
376      OrderStatus os ON so.StatusID = os.StatusID
377  JOIN
378      PaymentMode pm ON so.PaymentID = pm.PaymentID
379  JOIN
380      Product p ON so.ProductID = p.ProductID;
381
382  SELECT * FROM OrderDetails WHERE OrderID = 2;
```

Results Messages

	OrderID	CustomerName	EmployeeName	StoreName	OrderStatus	PaymentType	ProductName	ProductQty	ProductTotal	OrderTotal
1	2	Dwight Schrute	Jane Smith	Uptown Fitness	Completed	Credit Card	Resistance Bands	3	59.97	2609.76
2	2	Dwight Schrute	Jane Smith	Uptown Fitness	Completed	Credit Card	Running Shoes	7	629.93	2609.76
3	2	Dwight Schrute	Jane Smith	Uptown Fitness	Completed	Credit Card	Hiking Backpack	9	1169.91	2609.76
4	2	Dwight Schrute	Jane Smith	Uptown Fitness	Completed	Credit Card	Fitness Tracker	5	749.95	2609.76

VIEW- TOP 5 HOT SELLING PRODUCTS



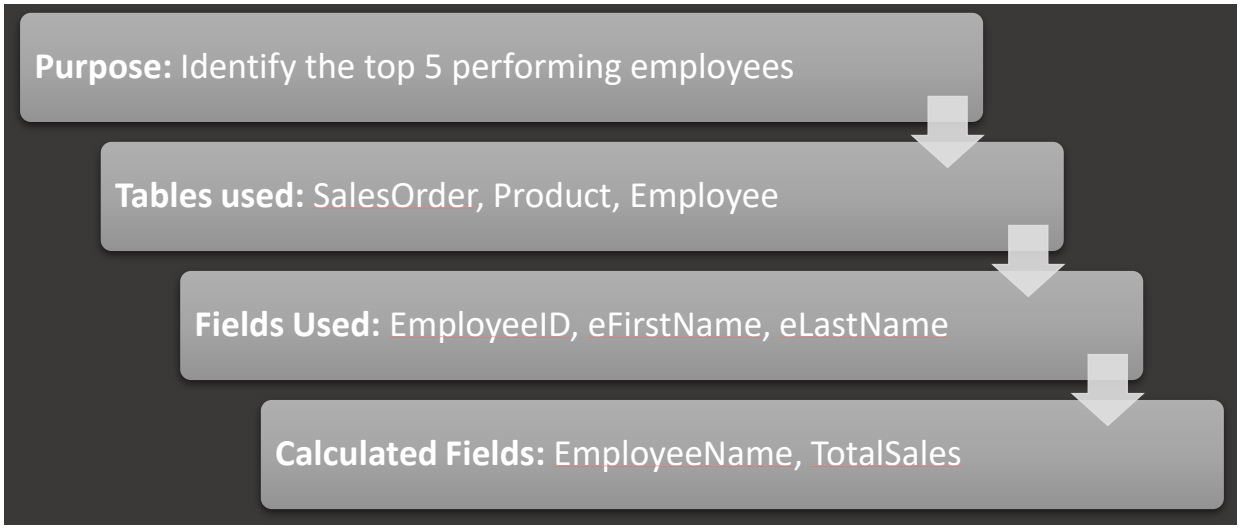
QUERIES

```
384   CREATE VIEW HotProducts AS
385   SELECT
386       p.ProductID,
387       p.ProductName,
388       SUM(so.ProductQty) AS TotalQuantitySold,
389       SUM(so.ProductQty * p.Price) AS TotalSales
390   FROM
391       SalesOrder so
392   JOIN
393       Product p ON so.ProductID = p.ProductID
394   GROUP BY
395       p.ProductID, p.ProductName
396   ORDER BY
397       TotalSales DESC
398   LIMIT 5;
399
400   SELECT * FROM HotProducts
```

Results Messages

	ProductID	ProductName	TotalQuantitySold	TotalSales
1	1	Treadmill	28	27999.72
2	10	Elliptical Machine	15	8999.85
3	7	Hiking Backpack	28	3639.72
4	8	Fitness Tracker	21	3149.79
5	2	Dumbbell Set	15	2999.85

VIEW- BEST PERFORMING EMPLOYEE



QUERIES

```
402    SELECT
403        e.EmployeeID,
404        CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,
405        SUM(so.ProductQty * p.Price) AS TotalSales
406    FROM
407        SalesOrder so
408    JOIN
409        Employee e ON so.EmployeeID = e.EmployeeID
410    JOIN
411        Product p ON so.ProductID = p.ProductID
412    GROUP BY
413        e.EmployeeID, EmployeeName
414    ORDER BY
415        TotalSales DESC
416    LIMIT 5;
```

Results Messages

	EmployeeID	EmployeeName	TotalSales
1	6	David Brown	13289.71
2	3	Mike Taylor	10319.78
3	8	Linda Garcia	8599.75
4	1	John Doe	5819.83
5	5	Sarah Lee	4489.71

Conclusion

In this article, we have delved into the entire process of designing a good database from start to finish, using Fitflex as our case study. By understanding and applying good design principles, efforts have been made to create a database that is robust, scalable, and aligned with business objectives. also establishing clear relationships between tables. By following these guidelines, one can create a database that does not only stores information efficiently but also enhances business operations and decision-making processes.

Recall that a well-designed database is a powerful tool that supports business growth and efficiency. By investing the time and effort into proper database design, data management processes are streamlined, accurate, and ready to support business needs both now and in the future.