

# Sales Management System

## Introduction

Sales management is a vital part of every business that involves tracking customer information, managing products, recording orders, and analyzing revenue. A well-designed system helps companies understand their sales performance, manage inventory, and provide better service to customers.

Traditional sales tracking methods such as spreadsheets or manual records are often time-consuming, prone to errors, and lack real-time updates. This can lead to issues like incorrect stock management, delayed order processing, and missed business opportunities.

To overcome these challenges, a **Sales Management System** using **MySQL** provides a digital way to organize and manage sales-related data. It ensures efficient handling of customers, products, and orders — all in one place.

This project demonstrates how a structured database can help store and retrieve information accurately, generate sales reports, and support business decision-making effectively.

## Objective

The main objective of this project is to design and implement a **Sales Management System** using **MySQL** to efficiently manage all information related to customers, products, and orders.

## The key objectives are:

- To store customer details such as name, city, and country.
- To manage product details including product line, quantity, and price.
- To record orders and order details for each customer.
- To analyze data for top-selling products and revenue generation.
- To provide accurate and reliable data storage for business analysis.

Through this project, we aim to show how a simple MySQL database can serve as a foundation for sales and inventory tracking in small to medium businesses.

## Scope

This project focuses on designing a small-scale database system to handle basic sales management operations. It includes:

- Managing **customer records**
- Tracking **products and stock levels**
- Recording **orders and sales details**
- Generating simple **sales reports**

It does not include advanced features such as web integration, billing automation, or data visualization dashboards. The goal is to show how **MySQL** can efficiently manage structured business data.

## Technology Used

- **Database:** MySQL
- **Concepts Used:** Table creation, Primary and Foreign Keys, Joins, Aggregate Functions, Grouping, Sorting, Filtering

MySQL provides reliable data storage, easy querying, and supports complex business operations like sales tracking and revenue analysis.

## Database Design and Tables

### Tables:

- **Customers:** Stores customer details (name, city, and country)
- **Products:** Stores product details (name, line, quantity, and price)
- **Orders:** Stores customer order information
- **Order\_Details:** Stores details of each ordered product (quantity and price per item)

# MySQL Implementation

## 1. Table Creation Scripts

-- Customers Table

```
create table customers(  
    customer_id varchar(10) PRIMARY KEY,  
    customer_name varchar(20),  
    city varchar(15),  
    country varchar(15)  
);
```

-- Products Table

```
create table products (  
    product_id VARCHAR(10) PRIMARY KEY,  
    product_name VARCHAR(50),  
    product_line VARCHAR(50),  
    buy_price DECIMAL(10,2),  
    msrp DECIMAL(10,2),  
    quantity_in_stock INT  
);
```

-- Orders Table

```
create table orders(  
    order_id varchar(10) PRIMARY KEY,  
    customer_id varchar(10),  
    order_date date,  
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)  
);
```

-- Order Details Table

```
create table order_details (  
    order_id varchar(10),  
    product_id varchar(10),  
    quantity int,  
    price_each decimal(10,2),  
    PRIMARY KEY(order_id, product_id),  
    FOREIGN KEY(order_id) REFERENCES orders(order_id),  
    FOREIGN KEY(product_id) REFERENCES products(product_id)  
);
```

## 2. Sample Data Insertion

-- Insert into Customers

insert into customers values

('C01','Smith','New York','USA'),  
('C02','Lee','Los Angeles','USA'),  
('C03','David','London','UK'),  
('C04','Emma','Berlin','Germany'),  
('C05','Johnson','Sydney','Australia'),  
('C06','Martin','Toronto','Canada');

-- Insert into Products

insert into products values

('P001','1957 Chevy Pickup','Classic Cars', 20.00, 40.00, 100),  
('P002','1965 Aston Martin DB5','Classic Cars', 25.00, 50.00, 80),  
('P003','1958 Corvette Limited','Classic Cars', 22.00, 45.00, 90),  
('P004','1969 Ford Falcon','Vintage Cars', 18.00, 35.00, 70),  
('P005','1972 Alpine Renault 1600s','Vintage Cars', 20.00, 38.00, 60),  
('P006','Harley Davidson Bike','Motorcycles', 15.00, 30.00, 50),  
('P007','1961 Jaguar E-Type','Classic Cars', 23.00, 48.00, 40);

-- Insert into Orders

insert into orders values

('O001','C02','2025-02-04'),  
('O002','C04','2025-02-10'),  
('O003','C02','2025-03-19'),  
('O004','C05','2025-04-27'),  
('O005','C01','2025-05-05'),  
('O006','C06','2025-05-18'),  
('O007','C01','2025-06-01'),  
('O008','C06','2025-06-16');

-- Insert into Order\_Details

insert into order\_details values

('O001', 'P001', 2, 40.00),  
('O001', 'P002', 1, 50.00),  
('O002', 'P003', 3, 45.00),  
('O003', 'P004', 2, 35.00),  
('O004', 'P005', 1, 38.00),  
('O005', 'P006', 4, 30.00);

-- Select to verify

select \* from customers;

select \* from products;

select \* from orders;

select \* from order\_details;

### 3. Queries for Real Use Cases

-- 1.Find Product Line Generating the Most Revenue

select p.product\_line,

sum(od.quantity \* od.price\_each) as total\_Revenue

from order\_details od

join products p on od.product\_id=p.product\_id

group by p.product\_line

order by total\_Revenue desc

limit 1;

-- 2.Identify Top-Selling Products

select product\_name,

sum(od.quantity) as Top\_Selling\_Products

from order\_details od

join products p on od.product\_id=p.product\_id

group by p.product\_name

order by Top\_Selling\_Products desc;



-- 3. Customers with the Most Orders

select

c.customer\_id,

c.customer\_name,

COUNT(o.order\_id) AS total\_orders

from customers c

join orders o ON c.customer\_id = o.customer\_id

group by c.customer\_id, c.customer\_name

order by total\_orders desc

limit 5;

select \* from products;

-- 4. Products with Low Inventory not work

select product\_id,

product\_name,

quantity\_in\_stock from products

where quantity\_in\_stock < 60;

-- 5.Find the average order value per customer

select c.customer\_name,

ROUND(AVG(od.quantity \* od.price\_each), 2) AS  
average\_order\_value from customers c

join orders o on

c.customer\_id = o.customer\_id

join order\_details od

ON o.order\_id = od.order\_id

group by c.customer\_name

order by average\_order\_value DESC;

-- 6.Find Customers Who Haven't Placed Any Orders

select

c.customer\_id,

c.customer\_name,

c.city,

c.country

from customers c

LEFT JOIN orders o

ON c.customer\_id = o.customer\_id

where o.order\_id IS NULL;

-- 7.Product with Highest Stock Value

```
select product_name,  
quantity_in_stock * buy_price AS total_stock_value  
from products  
order by total_stock_value DESC  
limit 1;
```

-- 8.Find the Most Popular Product Line

```
select  
    p.product_line,  
    SUM(od.quantity) AS total_sold  
from order_details od  
JOIN products p ON od.product_id = p.product_id  
group by p.product_line  
order by total_sold DESC  
limit 1;
```

Output:

```
select * from customers;
```

customer_id	customer_name	city	country
C01	Smith	New York	USA
C02	Lee	Los Angeles	USA
C03	David	London	UK
C04	Pena	Berlin	Germany
C05	Johnson	Sydney	Australia
C06	Martin	Toronto	Canada

```
select * from products;
```

product_id	product_name	product_line	buy_price	msrp	quantity_in_stock
P001	1957 Chevy Pickup	Classic Cars	20.00	40.00	100
P002	1965 Aston Martin DB5	Classic Cars	25.00	50.00	80
P003	1958 Corvette Limited	Classic Cars	22.00	45.00	90
P004	1969 Ford Falcon	Vintage Cars	18.00	35.00	70
P005	1972 Alpine Renault 1600s	Vintage Cars	20.00	38.00	60
P006	Harley Davidson Bike	Motorcycles	15.00	30.00	50
P007	1961 Jaguar E-Type	Classic Cars	23.00	48.00	40

```
select * from orders;
```

order_id	customer_id	order_date
O001	C02	2023-02-04
O002	C04	2023-02-10
O003	C02	2023-03-18
O004	C05	2023-04-27
O005	C01	2023-05-05
O006	C06	2023-05-15
O007	C01	2023-06-01
O008	C06	2023-06-16

```
select * from order_details;
```

order_id	product_id	quantity	price_each
O001	P001	2	40.00
O001	P002	1	50.00
O002	P003	3	45.00
O003	P004	2	35.00
O004	P005	1	38.00
O005	P006	4	30.00

## Find Product Line Generating the Most Revenue

product_line	total_revenue
Classic Cars	249.00

## Identify Top-Selling Products

product_name	Top_Selling_Products
Harley Davidson Bike	4
1958 Corvette Limited	3
1957 Chevy Pickup	2
1969 Ford Falcon	2
1965 Aston Martin DB5	1
1972 Alpine Renault 1600s	1

## Customers with the Most Orders

customer_id	customer_name	total_orders
C01	Smith	2
C02	Lee	2
C06	Martin	2
C04	Emma	1

select \* from products;

product_id	product_name	product_line	buy_price	msrp	quantity_in_stock
P001	1957 Chevy Pickup	Classic Cars	20.00	40.00	100
P002	1965 Aston Martin DB5	Classic Cars	25.00	50.00	80
P003	1958 Corvette Limited	Classic Cars	22.00	45.00	90
P004	1969 Ford Falcon	Vintage Cars	18.00	35.00	70
P005	1972 Alpine Renault 1600s	Vintage Cars	20.00	38.00	60
P006	Harley Davidson Bike	Motorcycles	15.00	30.00	50

## Products with Low Inventory not work

product_id	product_name	quantity_in_stock
P006	Harley Davidson Bike	50
P007	1961 Jaguar E-Type	40

## Find the average order value per customer

customer_name	average_order_value
Emma	135.00
Smith	120.00
Lee	66.67
Johnson	38.00

## Find Customers Who Haven't Placed Any Order

customer_id	customer_name	city	country
C03	David	London	UK

## Product with Highest Stock Value

product_name	total_stock_value
1965 Aston Martin DB5	2000.00

## Find the Most Popular Product Line

product_line	total_sold
Classic Cars	6

## **Live SQL Demo:**

The database schema, sample data, and queries for project have been Implemented and tested online. You can view and run them directly Using the following link:

<https://sqlfiddle.com/mysql/online-compiler?id=b8895a47-b2e8-4d27-b8fb-bc3f2b40a74a>

## **Conclusion**

The **Sales Management System** efficiently manages customers, products, and order data using **MySQL**.

It helps track sales, monitor stock levels, and analyze revenue. This digital solution minimizes manual errors, ensures data consistency, and supports better decision-making.

With further enhancements such as a user interface or real-time dashboards, this system can be expanded into a complete business sales solution.