Problem Definition

Business Context

Medical Health Company is a private pharmaceutical retail company operating in the healthcare industry. The company sells medical drugs across four regional branches located in Kigali, Huye, Gisenyi, and Musanze. This analysis is conducted within the Information Technology and Sales Analytics departments to support data-driven decision-making.

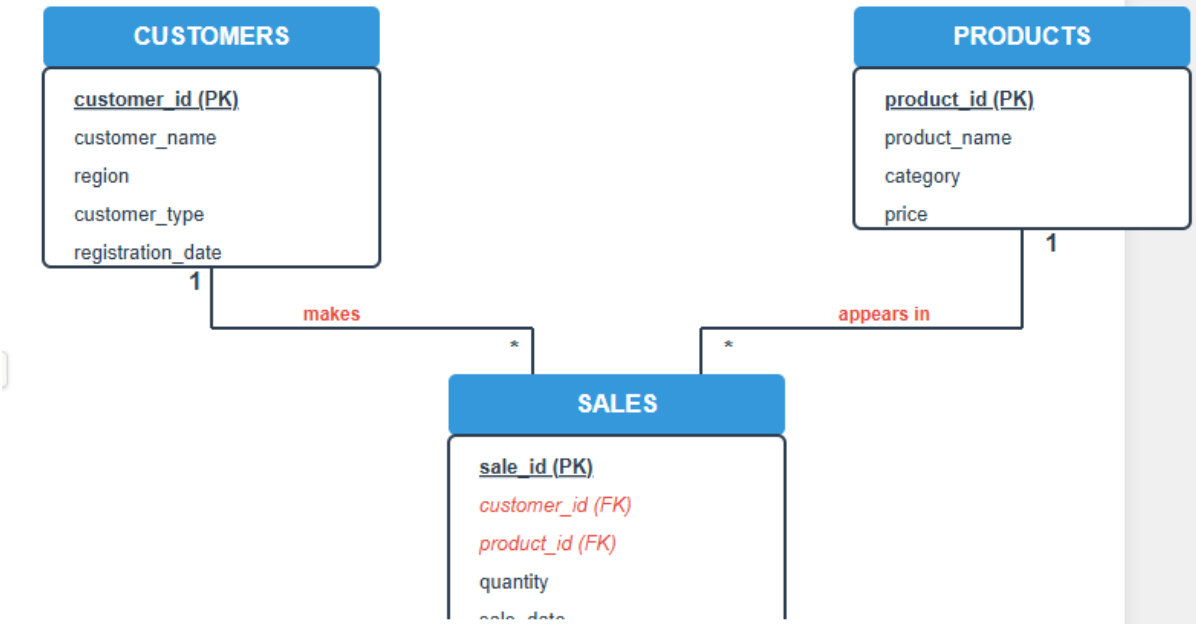
Data Challenge

Medical Health Company records customer transactions, product sales, and regional information in separate database tables. However, management currently lacks analytical visibility to determine top-selling drugs by region, understand customer purchasing frequency, and effectively group customers for targeted marketing campaigns. Manual reporting methods are inefficient and do not support advanced sales analysis.

Expected Outcome

The objective is to generate actionable insights that identify top products per region, measure customer purchase behavior, and segment customers into meaningful groups. These insights will support targeted marketing strategies, inventory optimization, and improved regional sales performance through the use of SQL JOINs and window functions.

Database Schema Design



Legend & Relationships

PK = Primary Key (underlined)
FK = Foreign Key (italic, red text)

CREATE TABLES

```
-- TABLE 3: Sales
-- Purpose: Store transaction records

CREATE TABLE Sales (
    sale_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    quantity INT,
    sale_date DATE,
    total_amount DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

```sql
-- TABLE 2: Products
-- Purpose: Store product catalog

CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10,2) NOT NULL
);

-- TABLE 1: Customers
-- Purpose: Store customer information

CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100) NOT NULL,
    region VARCHAR(50) NOT NULL,
    customer_type VARCHAR(50),
    registration_date DATE
);
```

SQL JOIN

INNER JOIN

sql

```sql
SELECT
    s.sale_id,
    c.customer_name,
    c.region,
    p.product_name,
    p.category,
    s.quantity,
    s.sale_date,
    s.total_amount
FROM Sales s
INNER JOIN Customers c ON s.customer_id = c.customer_id
INNER JOIN Products p ON s.product_id = p.product_id
ORDER BY s.sale_date DESC
LIMIT 15;
```

LEFT JOIN

```sql
SELECT
    c.customer_id,
    c.customer_name,
    c.region,
    c.customer_type,
    c.registration_date,
    COUNT(s.sale_id) as total_sales
FROM Customers c
LEFT JOIN Sales s ON c.customer_id = s.customer_id
GROUP BY c.customer_id, c.customer_name, c.region, c.customer_type, c.registration_d
HAVING COUNT(s.sale_id) = 0
ORDER BY c.registration_date DESC;
```

RIGHT JOIN

```sql
SELECT
    p.product_id,
    p.product_name,
    p.category,
    p.price,
    COUNT(s.sale_id) as times_sold,
    COALESCE(SUM(s.total_amount), 0) as total_revenue
FROM Sales s
RIGHT JOIN Products p ON s.product_id = p.product_id
GROUP BY p.product_id, p.product_name, p.category, p.price
HAVING COUNT(s.sale_id) = 0
ORDER BY p.price DESC;
```

FULL OUTER JOIN

```sql
SELECT
    COALESCE(c.customer_name, 'No Customer') as customer,
    COALESCE(p.product_name, 'No Product') as product,
    COALESCE(s.total_amount, 0) as amount
FROM Customers c
FULL OUTER JOIN Sales s ON c.customer_id = s.customer_id
FULL OUTER JOIN Products p ON s.product_id = p.product_id
WHERE c.customer_id IS NULL OR p.product_id IS NULL OR s.sale_id IS NULL
ORDER BY customer, product;
```

SELF JOIN

```sql
SELECT
    c1.customer_id as customer1_id,
    c1.customer_name as customer1_name,
    c1.customer_type as customer1_type,
    c2.customer_id as customer2_id,
    c2.customer_name as customer2_name,
    c2.customer_type as customer2_type,
    c1.region as shared_region
FROM Customers c1
INNER JOIN Customers c2
    ON c1.region = c2.region
    AND c1.customer_id < c2.customer_id
ORDER BY c1.region, c1.customer_name;
```

WINDOWS FUNCTION

```sql
SELECT
    region,
    EXTRACT(MONTH FROM sale_date) as month,
    COUNT(sale_id) as sales_count,
    SUM(total_amount) as regional_revenue,
    RANK() OVER (
        PARTITION BY EXTRACT(MONTH FROM sale_date)
        ORDER BY SUM(total_amount) DESC
    ) as region_rank_in_month,
    SUM(total_amount) * 100.0 / SUM(SUM(total_amount)) OVER (
        PARTITION BY EXTRACT(MONTH FROM sale_date)
    ) as pct_of_monthly_revenue
FROM Sales s
INNER JOIN Customers c ON s.customer_id = c.customer_id
GROUP BY region, EXTRACT(MONTH FROM sale_date)
ORDER BY month, regional_revenue DESC;
```



```sql
SELECT
    c.customer_name,
    EXTRACT(MONTH FROM s.sale_date) as month,
    SUM(s.total_amount) as monthly_revenue,
    SUM(SUM(s.total_amount)) OVER (
        PARTITION BY c.customer_id
        ORDER BY EXTRACT(MONTH FROM s.sale_date)
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) as ytd_revenue,
    COUNT(s.sale_id) as monthly_orders,
    SUM(COUNT(s.sale_id)) OVER (
        PARTITION BY c.customer_id
        ORDER BY EXTRACT(MONTH FROM s.sale_date)
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) as ytd_orders
FROM Sales s
INNER JOIN Customers c ON s.customer_id = c.customer_id
GROUP BY c.customer_id, c.customer_name, EXTRACT(MONTH FROM s.sale_date)
ORDER BY c.customer_name, month;
```

Results Analysis

Descriptive

Top-performing students and high-performing courses were identified.

Diagnostic

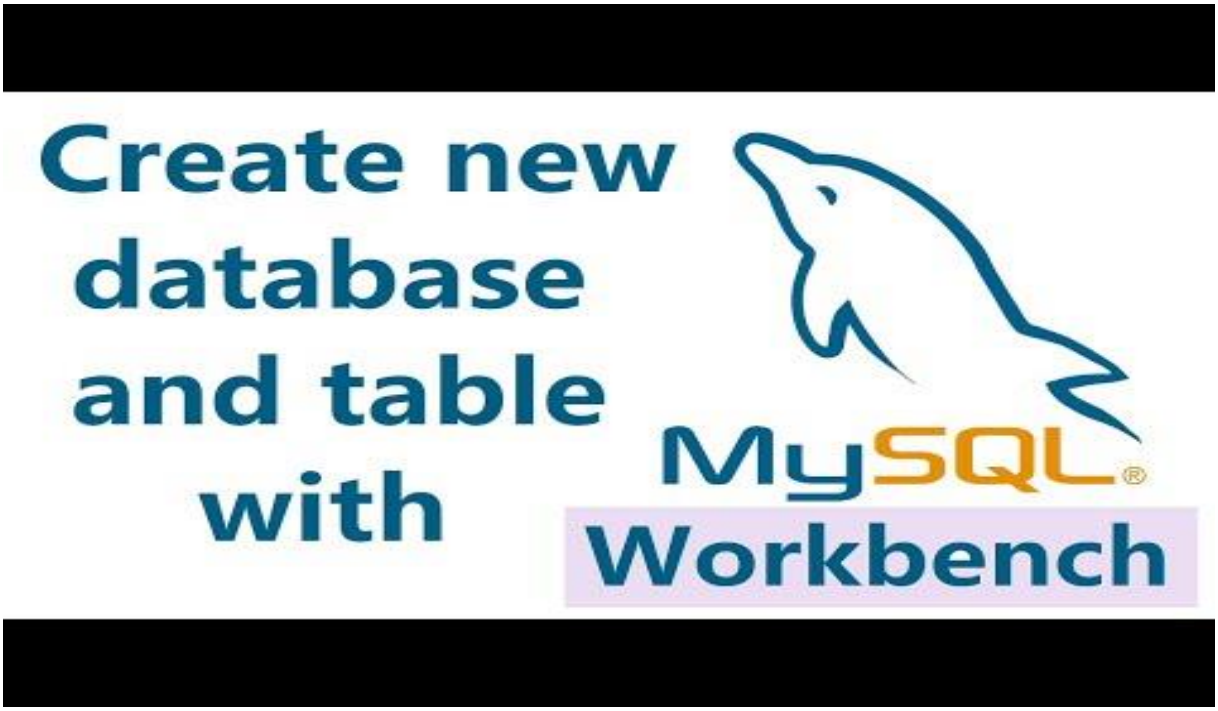
Performance differences are influenced by course difficulty and student engagement.

Prescriptive

Provide academic support to low-performing students and review course content where averages are low.

References

https://www.youtube.com/watch?v=wALCw0F8e9M&list=PPSV



https://www.youtube.com/watch?v=G3UAxg1cy8&list=PPSV