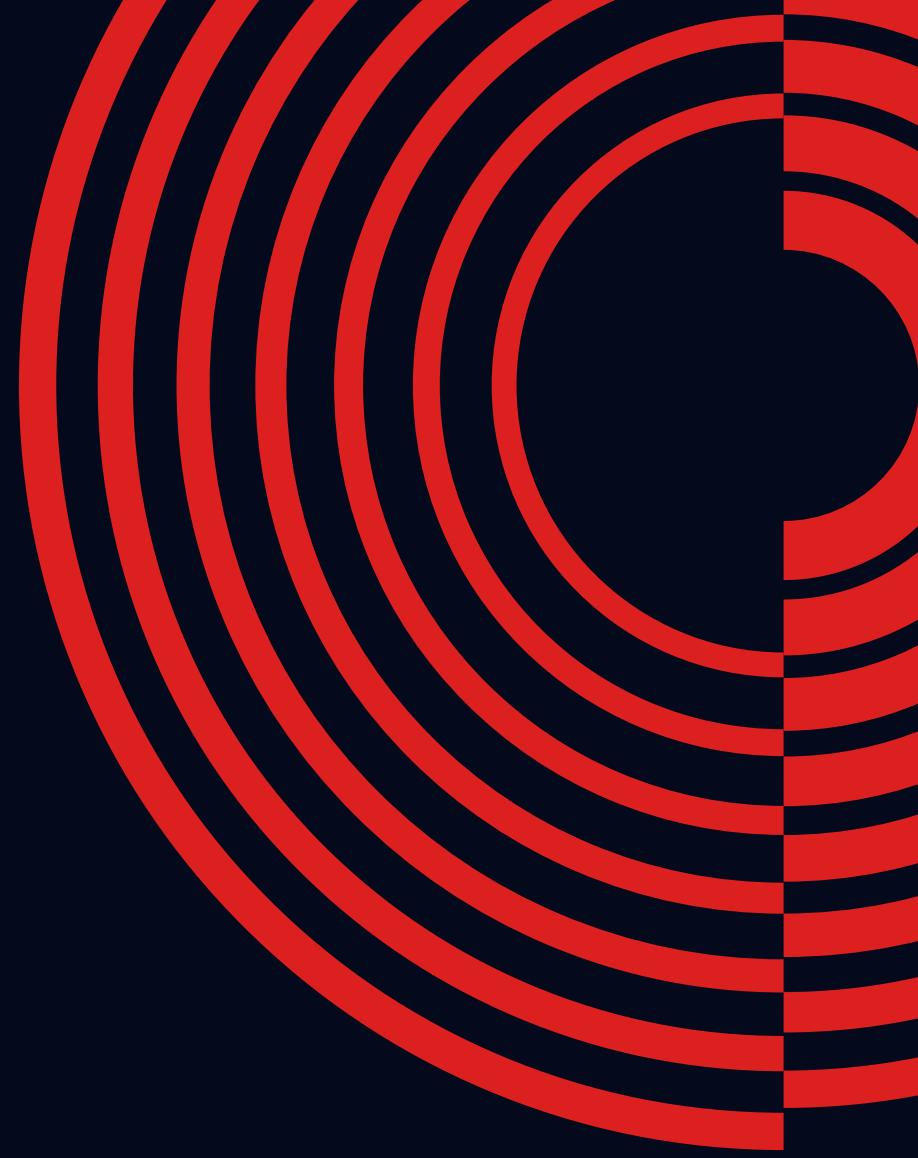


# 7



DATA ANALYTICS JOURNEY

# Gross Monthly Sales Report Using SQL



**FAISAL SHAHZAD**  
DATA ANALYST



# 1

## Introduction & Requirement

### Gross Monthly Sales Report Using SQL

Built a SQL-based report to calculate gross monthly sales for a specific customer.

#### **Business requirement:**

- Show sales date (monthly view)
- Calculate total gross sales amount

Purpose:

To track customer revenue contribution over time using accurate pricing logic.

# 2

## Revenue Logic

### How Gross Sales Are Calculated

Gross Sales = Sold Quantity × Gross Price

Pricing is joined using:

→ Product code

→ Fiscal year derived from sales date

This ensures pricing accuracy

# 3

## Fiscal Year Function Code

```
-- BLOCK 1: Get raw sales data WITH january_sales AS ( SELECT product_id, SUM(quantity)
1 CREATE FUNCTION get_fiscal_year (calendar_date DATE)
2 RETURNS INTEGER
3 DETERMINISTIC
4 BEGIN
5     -- Declare a variable to store the fiscal year
6     DECLARE fiscal_year INT;
7
8     -- Add 4 months to the given calendar date
9     -- This shifts Jan-Mar dates into the correct fiscal year
10    SET fiscal_year = YEAR(DATE_ADD(calendar_date, INTERVAL 4 MONTH));
11
12    -- Return the calculated fiscal year
13    RETURN fiscal_year;
14 END;
15
```

Logic:

Calendar Date → +4 Months → Extract Year → Fiscal Year

## 4

# SQL Query Used

```
● ● ● -- BLOCK 1: Get raw sales data WITH january_sales AS ( SELECT product_id, SUM(quantity

1  -- Gross monthly sales report using fiscal year-aligned pricing
2  SELECT
3      s.date,
4      ROUND(SUM(g.gross_price * s.sold_quantity), 2) AS gross_price_total
5  FROM fact_sales_monthly s
6
7  -- Join pricing table using product code and fiscal year derived from
   date
8  JOIN fact_gross_price g
9      ON s.product_code = g.product_code
10     AND g.fiscal_year = get_fiscal_year(s.date)
11
12 -- Filter data for a specific customer
13 WHERE s.customer_code = 90002002
14
15 -- Aggregate gross sales at date (monthly) level
16 GROUP BY s.date
17
18 -- Show latest dates first
19 ORDER BY s.date DESC;
20
```



# 5

## Why This Works & Final Output

- Reuses the custom `get_fiscal_year()` function built in Post 1
- Ensures sales and pricing are aligned to the same fiscal year
- Aggregates gross revenue at the date (monthly) level

### **Final Output:**

- Date
- Total Gross Sales

The final result is exported to Excel for reporting and trend analysis.

SQL scripts and the Excel file are available in the GitHub repository.

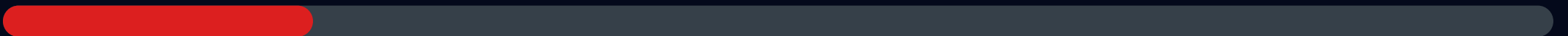
# 6

## Skills Demonstrated

- SQL joins and aggregation
- Revenue calculation
- Fiscal year handling
- Query reusability
- Business-focused reporting

### Question

How do you ensure pricing accuracy in your SQL revenue reports?



# 7

## Project Progression

### **Project Progression**

- Post 1: Built a reusable fiscal year function to handle Apr–Mar reporting
  - Post 2: Reused the same function to calculate monthly gross sales accurately
  - Ensured pricing and sales data remained aligned across reports
  - Followed a modular, scalable SQL design approach
  - Next: Monthly trends, customer performance, and deeper revenue analysis

### **Why this helps:**

Demonstrates structured learning, query reusability, and real-world analytics workflow.





FAISAL SHAHZAD

**Thanks!**

Follow me for another tips

DATA ANALYSTS