GenAI-Assisted Data Migration & BI

Project Execution Report

Generated on: 2025-09-28 15:38:25

Table of Contents

Schema Generation

Execution Step

Input:

Generate Drop and DDL statements for all tables.

Sample data from CUSTOMERS.csv:

customer_id,customer_name,address,phone_number,email,join_date

1,Jordan Smith,"859 Example Street, Chennai, TN",9395310485,jordan.smith@example.com,2024-03-12

2,Morgan Williams,"854 Example Street, Chennai, TN",9210053353,morgan.williams@example.com,2021-10-10

3,Cameron Johnson,"704 Example Street, Chennai, TN",9553035110,cameron.johnson@example.com,2025-05-22

4,Alex Johnson,"323 Example Street, Chennai, TN",9349817734,alex.johnson@example.com,2022-09-26

5,Dakota Smith,"674 Example Street, Chennai, TN",9313500298,dakota.smith@example.com,2021-07-21

Sample data from INVENTORY.csv:

product_id,product_name,category,quantity_in_stock,price_per_unit

1,Product_1,Electronics,152,137.66

2,Product_2,Grocery,485,92.85

3,Product_3,Furniture,398,7.59

4,Product_4,Electronics,320,188.73

5,Product_5,Toys,156,114.84

Sample data from SALES.csv:

sale_id,customer_id,product_id,quantity,sale_date,total_amount

1,68,1,9,2025-03-24,1238.94

2,14,9,5,2025-06-26,260.0

3,14,48,9,2025-06-06,1670.67

4,35,19,10,2025-05-09,1157.0

5,44,14,5,2024-12-09,406.3

Output/Result:

```
DROP TABLE IF EXISTS SALES;
DROP TABLE IF EXISTS CUSTOMERS;
DROP TABLE IF EXISTS INVENTORY;

CREATE TABLE CUSTOMERS (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    address VARCHAR(255),
    phone_number VARCHAR(20),
    email VARCHAR(255),
    join_date DATE
);

CREATE TABLE INVENTORY (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(255),
    category VARCHAR(100),
    quantity_in_stock INT,
    price_per_unit DECIMAL(10, 2)
);

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 INVENTORY(product_id)
```

);

Query & Logic Translation

Execution Step

Input:

```
-- Procedure to get monthly sales
CREATE OR REPLACE PROCEDURE GetMonthlySales(p_month IN NUMBER, p_year IN NUMBER,
result OUT SYS_REFCURSOR)
IS
BEGIN
    OPEN result FOR
    SELECT TO_CHAR(sale_date, 'YYYY-MM') AS sale_month,
           SUM(total_amount) AS total_sales
    FROM SALES
    WHERE EXTRACT(MONTH FROM sale_date) = p_month
      AND EXTRACT(YEAR FROM sale_date) = p_year
    GROUP BY TO_CHAR(sale_date, 'YYYY-MM');
END GetMonthlySales;
/

-- Function to check reorder point for inventory
CREATE OR REPLACE FUNCTION NeedReorder(p_product_id IN NUMBER) RETURN BOOLEAN
IS
    qty NUMBER;
BEGIN
    SELECT quantity_in_stock INTO qty
    FROM INVENTORY
    WHERE product_id = p_product_id;
    IF qty < 100 THEN
       RETURN TRUE;
    ELSE
       RETURN FALSE;
    END IF;
```

```
END NeedReorder;
/

-- Sample business query: Get top 5 customers by total purchase
SELECT c.customer_name, SUM(s.total_amount) AS total_purchase
FROM SALES s
JOIN CUSTOMERS c ON s.customer_id = c.customer_id
GROUP BY c.customer_name
ORDER BY total_purchase DESC
FETCH FIRST 5 ROWS ONLY;
/
```

Output/Result:

```
CREATE PROCEDURE GetMonthlySales(IN p_month INT, IN p_year INT)
BEGIN
    SELECT DATE_FORMAT(sale_date, '%Y-%m') AS sale_month,
           SUM(total_amount) AS total_sales
    FROM SALES
    WHERE MONTH(sale_date) = p_month
      AND YEAR(sale_date) = p_year
    GROUP BY DATE_FORMAT(sale_date, '%Y-%m');
END;

CREATE FUNCTION NeedReorder(p_product_id INT) RETURNS BOOLEAN
READS SQL DATA
BEGIN
    DECLARE qty INT;
    SELECT quantity_in_stock INTO qty
    FROM INVENTORY
    WHERE product_id = p_product_id;
    IF qty < 100 THEN
       RETURN TRUE;
    ELSE
       RETURN FALSE;
```

```
    END IF;
END;

SELECT c.customer_name, SUM(s.total_amount) AS total_purchase
FROM SALES s
JOIN CUSTOMERS c ON s.customer_id = c.customer_id
GROUP BY c.customer_name
ORDER BY total_purchase DESC
LIMIT 5;
```

Automated Testing

Execution Step

Input:

Find number of sales for each customer

Output/Result:

```
SELECT C.customer_name, COUNT(S.sale_id) AS number_of_sales
FROM CUSTOMERS C
JOIN SALES S ON C.customer_id = S.customer_id
GROUP BY C.customer_name
ORDER BY number_of_sales DESC;
```

BI Reporting & Visualization

Execution Step

Input:

Show monthly sales totals for the past year. Include month, total quantity sold, and total amount.

Output/Result:

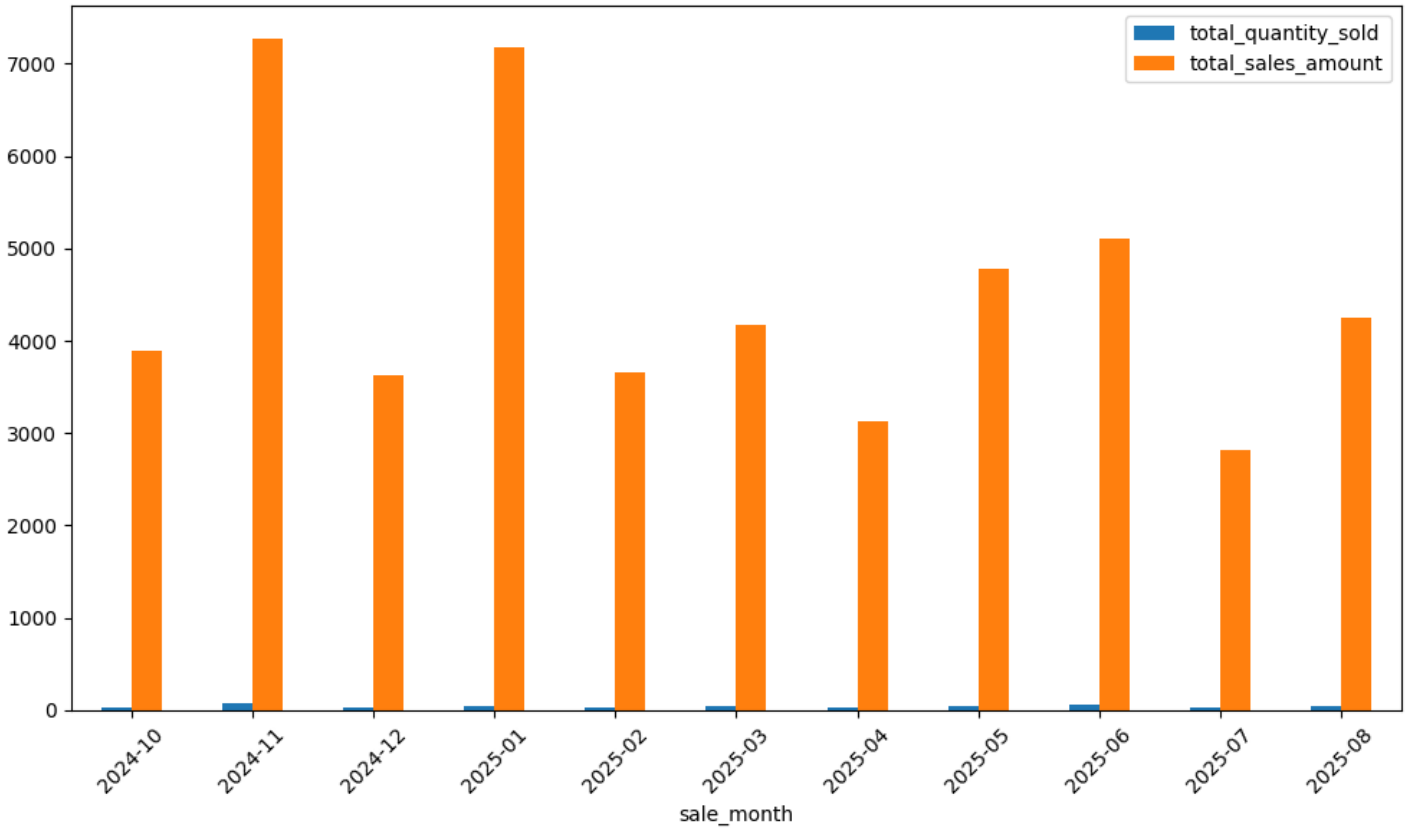
```
SELECT
  DATE_FORMAT(sale_date, '%Y-%m') AS sale_month,
  SUM(quantity) AS total_quantity_sold,
  SUM(total_amount) AS total_sales_amount
FROM SALES
WHERE
  sale_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
GROUP BY
  DATE_FORMAT(sale_date, '%Y-%m')
ORDER BY
  sale_month;
```

Query Results

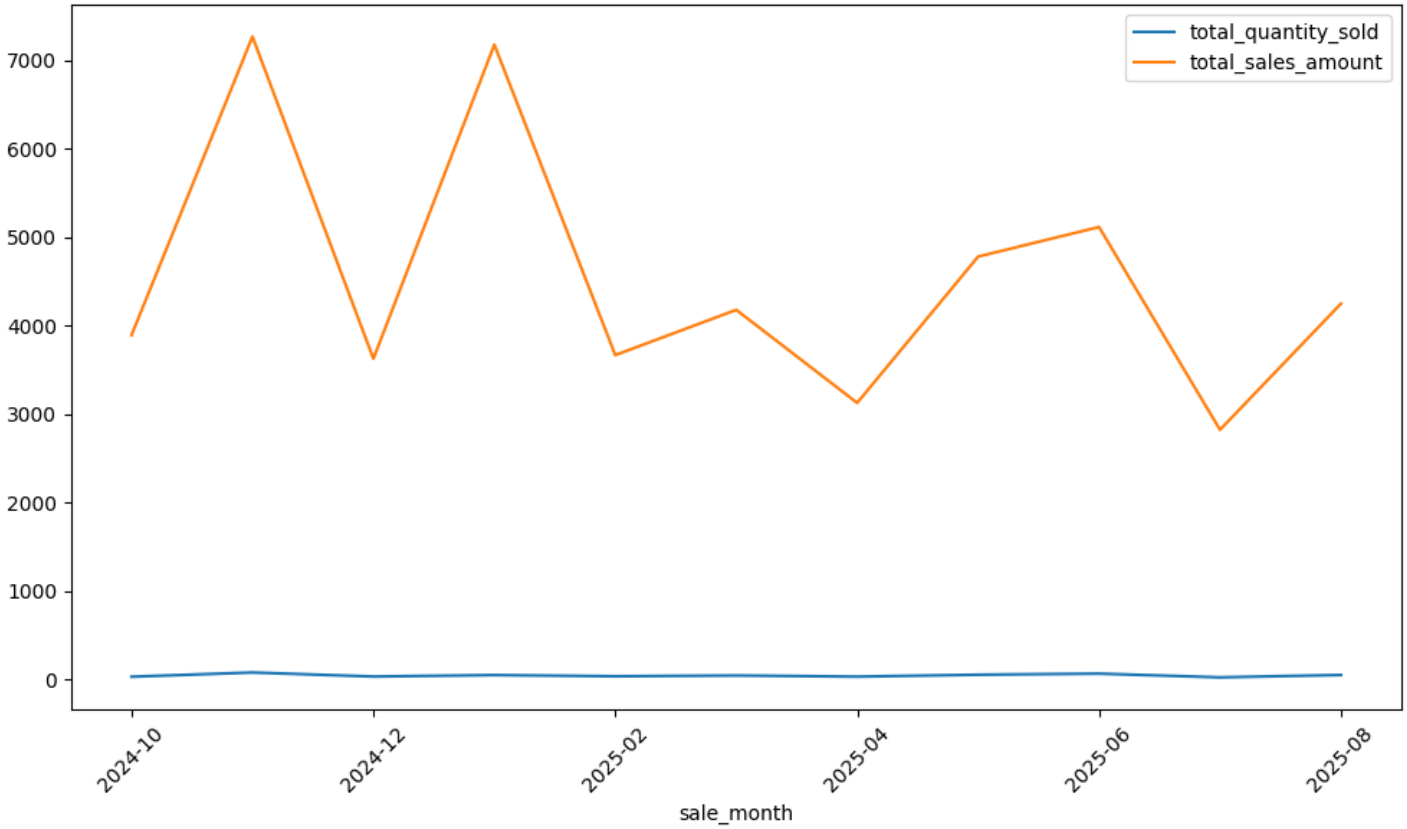
sale_month	total_quantity_sold	total_sales_amount
2024-10	30.0	3891.81
2024-11	78.0	7265.88
2024-12	32.0	3624.78
2025-01	49.0	7174.83
2025-02	35.0	3665.75
2025-03	44.0	4177.15
2025-04	31.0	3125.46
2025-05	53.0	4779.09
2025-06	65.0	5113.53
2025-07	23.0	2821.53
2025-08	50.0	4247.02

Visualizations

Bar Chart



Line Chart


total_quantity_sold
total_sales_amount
7000
6000
5000
4000
3000
2000
1000
0
2024-10
2024-12
2025-02
2025-04
2025-06
2025-08
sale_month