GenAI-Assisted Data Migration Report

Run Timestamp: 2025-09-18 03:11:18.464811

schema_sql

~~~
```sql
CREATE TABLE CUSTOMERS (
  customer_id INT PRIMARY KEY,
  customer_name VARCHAR(255) NOT NULL,
  address VARCHAR(255) NOT NULL,
  phone_number INT NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  join_date DATE NOT NULL
);

CREATE TABLE INVENTORY (
  product_id INT PRIMARY KEY,
  product_name VARCHAR(255) NOT NULL,
  category VARCHAR(255) NOT NULL,
  quantity_in_stock INT NOT NULL DEFAULT 0,
  price_per_unit DECIMAL(10, 2) NOT NULL
);

CREATE TABLE SALES (
  sale_id INT PRIMARY KEY,
  customer_id INT NOT NULL,
  product_id INT NOT NULL,
  quantity INT NOT NULL DEFAULT 1,
  sale_date DATE NOT NULL,
  total_amount DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (customer_id) REFERENCES CUSTOMERS(customer_id),
  FOREIGN KEY (product_id) REFERENCES INVENTORY(product_id)
);
~~~

~~~
## validation_sql

```sql
```sql
-- 1. Count rows in CUSTOMERS, INVENTORY, SALES
SELECT
    (SELECT COUNT(*) FROM CUSTOMERS) AS customers_count,
    (SELECT COUNT(*) FROM INVENTORY) AS inventory_count,
    (SELECT COUNT(*) FROM SALES) AS sales_count;

-- 2. Verify every SALES.customer_id exists in CUSTOMERS
SELECT
    s.customer_id
FROM
    SALES s
LEFT JOIN
    CUSTOMERS c ON s.customer_id = c.customer_id
WHERE
    c.customer_id IS NULL;

-- 3. Verify every SALES.product_id exists in INVENTORY
~~~

```
SELECT
    s.product_id
FROM
    SALES s
LEFT JOIN
    INVENTORY i ON s.product_id = i.product_id
WHERE
    i.product_id IS NULL;

-- 4. Total of SALES.total_amount
SELECT
    SUM(total_amount) AS total_sales
FROM
    SALES;
```

~~~
## validation_results

[
  {
    "query": "```sql\n-- 1. Count rows in CUSTOMERS, INVENTORY, SALES\nSELECT \n    (SELECT COUNT(*) FROM CUSTOMERS) AS
customers_count,\n    (SELECT COUNT(*) FROM INVENTORY) AS inventory_count,\n    (SELECT COUNT(*) FROM SALES) AS
sales_count",
    "error": "1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server
version for the right syntax to use near '```sql\n-- 1. Count rows in CUSTOMERS, INVENTORY, SALES\nSELECT \n    (SELECT
COUNT' at line 1"
  },
  {
    "query": "-- 2. Verify every SALES.customer_id exists in CUSTOMERS\nSELECT \n    s.customer_id\nFROM \n    SALES
s\nLEFT JOIN \n    CUSTOMERS c ON s.customer_id = c.customer_id\nWHERE \n    c.customer_id IS NULL",
    "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist"
  },
  {
    "query": "-- 3. Verify every SALES.product_id exists in INVENTORY\nSELECT \n    s.product_id\nFROM \n    SALES
s\nLEFT JOIN \n    INVENTORY i ON s.product_id = i.product_id\nWHERE \n    i.product_id IS NULL",
    "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist"
  },
  {
    "query": "-- 4. Total of SALES.total_amount\nSELECT \n    SUM(total_amount) AS total_sales\nFROM \n    SALES",
    "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist"
  },
  {
    "query": "```",
    "error": "1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server
version for the right syntax to use near '```' at line 1"
  }
]

## translated_sql

```sql
### MySQL Equivalent Procedures and Functions

Below are the equivalent MySQL stored procedures and functions for the provided Oracle PL/SQL procedures and functions.

#### Procedure to Get Monthly Sales

```sql
DELIMITER //
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//
DELIMITER ;
~~~

Note: MySQL does not support `OUT` parameters for stored procedures like Oracle does. Instead, the result set is returned directly. To call this procedure and retrieve the result set, you can use the following syntax:

```
CALL GetMonthlySales(1, 2022);
```

Function to Check Reorder Point for Inventory

```sql
DELIMITER //
CREATE FUNCTION NeedReorder(p_product_id INT) RETURNS BOOLEAN
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//
DELIMITER ;
```

Note: MySQL does not support `BOOLEAN` as a return type for stored functions. Instead, you can use `TINYINT(1)` or `INT` and return 0 or 1 to represent `FALSE` and `TRUE` respectively. However, in MySQL 8.0 and later, you can use the `BOOLEAN` type.

To call this function, you can use the following syntax:

```sql
SELECT NeedReorder(1);
```

Sample Business Query: Get Top 5 Customers by Total Purchase

```sql
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;
```

Note: MySQL uses the `LIMIT` clause to limit the number of rows returned, whereas Oracle uses the `FETCH FIRST` clause.

~~~

bi_sql

```sql
**KPI SQL Queries**
=====================

Below are the MySQL SQL queries for the requested KPIs:

### Monthly Sales Trend

This query will return the total sales for each month in the current year.

```sql
SELECT
    YEAR(order_date) AS year,
    MONTH(order_date) AS month,
    SUM(order_total) AS total_sales
FROM
    orders
WHERE
    YEAR(order_date) = YEAR(CURDATE())
GROUP BY
    YEAR(order_date),
    MONTH(order_date)
ORDER BY
    month;
~~~

## Top 5 Customers by Revenue

This query will return the top 5 customers with the highest total revenue.

```sql
SELECT
    c.customer_name,
    SUM(o.order_total) AS total_revenue
FROM
    orders o
```

```
JOIN
    customers c ON o.customer_id = c.customer_id
GROUP BY
    c.customer_name
ORDER BY
    total_revenue DESC
LIMIT 5;
```

Low Stock Products

This query will return all products with a quantity less than 100.

```
SELECT
    p.product_name,
    p.quantity
FROM
    products p
WHERE
    p.quantity < 100
ORDER BY
    p.quantity ASC;
```

Example Use Case

To use these queries, you would need to replace the table and column names with your actual database schema. For example, if your database has the following schema:

- `orders` table: `order_id`, `customer_id`, `order_date`, `order_total`
- `customers` table: `customer_id`, `customer_name`
- `products` table: `product_id`, `product_name`, `quantity`

You can run these queries in your MySQL database to get the desired KPIs.

Assumptions

- The `orders` table has a column `order_date` of type `date` or `datetime`.
- The `orders` table has a column `order_total` of type `decimal` or `float`.
- The `customers` table has a column `customer_name` of type `varchar`.
- The `products` table has a column `quantity` of type `int`.

Note

These queries assume that the database schema is as described above. You may need to modify the queries to fit your actual database schema. Additionally, these queries do not include any error handling or security measures, so you should modify them to fit your specific use case.