

anandjha90 / GenAI_LLM_Project

<

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

GenAI_LLM_Project / Capstone_Project / Output / migration_report_20250918_0319.md

anandjha90

Adding all the migration reports

da404ec · 4 minutes ago

184 lines (151 loc) · 4.72 KB

PreviewCodeBlame

Raw

GenAI-Assisted Data Migration Report

Run Timestamp: 2025-09-18 03:19:16.369717

schema_sql

```
CREATE TABLE customers (  
  customer_id BIGINT PRIMARY KEY,  
  customer_name VARCHAR(255) NOT NULL,  
  address VARCHAR(255) NOT NULL,  
  phone_number BIGINT NOT NULL,  
  email VARCHAR(255) UNIQUE NOT NULL,  
  join_date DATE NOT NULL  
);  
  
CREATE TABLE inventory (  
  product_id BIGINT PRIMARY KEY,  
  product_name VARCHAR(255) NOT NULL,  
  category VARCHAR(255) NOT NULL,  
  quantity_in_stock BIGINT NOT NULL DEFAULT 0,  
  price_per_unit DECIMAL(10, 2) NOT NULL  
);  
  
CREATE TABLE sales (  
  sale_id BIGINT PRIMARY KEY,  
  customer_id BIGINT NOT NULL,  
  product_id BIGINT NOT NULL,  
  quantity BIGINT 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

```
SELECT 'CUSTOMERS' AS table_name, COUNT(*) AS row_count FROM CUSTOMERS  
UNION ALL  
SELECT 'INVENTORY' AS table_name, COUNT(*) AS row_count FROM INVENTORY  
UNION ALL  
SELECT 'SALES' AS table_name, COUNT(*) AS row_count FROM SALES;  
  
SELECT * FROM SALES WHERE customer_id NOT IN (SELECT customer_id FROM CUSTOMERS);  
  
SELECT * FROM SALES WHERE product_id NOT IN (SELECT product_id FROM INVENTORY);  
  
SELECT SUM(total_amount) AS total_sales FROM SALES;
```

https://github.com/anandjha90/GenAI_LLM_Project/blob/main/Capstone_Project/Output/migration_report_20250918_0319.md

1/3

validation_results

```
[ { "query": "SELECT 'CUSTOMERS' AS table_name, COUNT() AS row_count FROM CUSTOMERS\nUNION ALL\nSELECT 'INVENTORY' AS table_name, COUNT() AS row_count FROM INVENTORY\nUNION ALL\nSELECT 'SALES' AS table_name, COUNT(*) AS row_count FROM SALES", "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist" }, { "query": "SELECT * FROM SALES WHERE customer_id NOT IN (SELECT customer_id FROM CUSTOMERS)", "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist" }, { "query": "SELECT * FROM SALES WHERE product_id NOT IN (SELECT product_id FROM INVENTORY)", "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist" }, { "query": "SELECT SUM(total_amount) AS total_sales FROM SALES", "error": "1146 (42S02): Table 'mysqldb.sales' doesn't exist" } ]
```

translated_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: In MySQL, we don't need to explicitly declare an OUT parameter for the result set. Instead, we can directly select the data in the procedure.

### Function to Check Reorder Point for Inventory

```
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 ;
```



### 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
LIMIT 5;
```



Note: In MySQL, we use the `LIMIT` clause instead of `FETCH FIRST` to limit the number of rows returned.

## Example Usage

To call the stored procedure and function, you can use the following queries:

```
-- Call the GetMonthlySales procedure
CALL GetMonthlySales(1, 2022);
```



```
-- Call the NeedReorder function
SELECT NeedReorder(1) AS need_reorder;
```

Make sure to replace the table and column names with the actual names in your database schema.



```
bi_sql

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

SELECT
    c.customer_name,
    SUM(oi.quantity * oi.unit_price) AS total_revenue
FROM
    customers c
JOIN
    orders o ON c.customer_id = o.customer_id
JOIN
    order_items oi ON o.order_id = oi.order_id
GROUP BY
    c.customer_name
ORDER BY
    total_revenue DESC
LIMIT 5;

SELECT
    p.product_name,
    p.quantity_in_stock
FROM
    products p
WHERE
    p.quantity_in_stock < 100;
```