BigQuery Routines - KirkYagami

BigQuery Routines encompass User-Defined Functions (UDFs), Table Functions, and SQL Stored Procedures, enabling the encapsulation of logic and repeated query patterns for reuse, organization, and optimization.

1. Managing Routines in BigQuery

Routines in BigQuery allow you to define reusable logic that can be executed within SQL queries. Managing these routines effectively involves:

- **Creation and Versioning**: Ensuring routines are appropriately named and versioned for maintainability.
- **Access Control**: Managing permissions to restrict or grant access to certain users or roles.
- **Documentation**: Providing clear documentation within the routine definition for ease of use and understanding.

Example: Creating and Managing a Simple SQL Stored Procedure

```
CREATE OR REPLACE PROCEDURE `bigdata3844.RevData.p_calculate_sales_metrics`()
BEGIN
  DECLARE total_sales FLOAT64 DEFAULT 0;
  DECLARE avg_sales FLOAT64 DEFAULT 0;

  -- Calculate total sales
  SET total_sales = (
    SELECT SUM(Weekly_Sales)
    FROM `bigdata3844.RevData.walmart`
  );

  -- Calculate average sales
  SET avg_sales = (
    SELECT AVG(Weekly_Sales)
    FROM `bigdata3844.RevData.walmart`
  );

  -- Output the results
  SELECT 'Total Sales' AS Metric, total_sales AS Value
  UNION ALL
  SELECT 'Average Sales', avg_sales;
END;
```

```
CALL `bigdata3844.RevData.p_calculate_sales_metrics`();
```

- **Versioning**: You might create a new version of the procedure for different fiscal years, e.g., `calculate_sales_metrics_v2024`.
- **Access Control**: You can set IAM roles to control who can run or modify the procedure.

2. User-Defined Functions (UDFs)

UDFs allow you to write custom scalar functions in **SQL or JavaScript** that can be invoked in SQL queries. They are powerful for extending BigQuery's functionality to meet specific needs.

Example: Creating a Scalar UDF to Classify Sales Performance

```
CREATE OR REPLACE FUNCTION
`bigdata3844.RevData.udfs_classify_sales`(weekly_sales FLOAT64)
RETURNS STRING
LANGUAGE js AS """
  if (weekly_sales > 1000000) {
    return 'High';
  } else if (weekly_sales > 500000) {
    return 'Medium';
  } else {
    return 'Low';
  }
""";

-- Using the UDF in a query
SELECT Store, Date, Weekly_Sales,
`bigdata3844.RevData.udfs_classify_sales`(Weekly_Sales) AS Sales_Performance
FROM `bigdata3844.RevData.walmart`;
```

- **Real-World Scenario**: This UDF can be used by analysts to quickly categorize sales data across multiple stores.

3. Table Functions

Table Functions return a table and can accept parameters. They are useful when you need to encapsulate complex queries that return tabular results.

Example: Creating a Table Function to Filter Sales Data

```
CREATE OR REPLACE TABLE FUNCTION `bigdata3844.RevData.filter_sales`(min_sales
FLOAT64)
RETURNS TABLE<Store INT64, Date DATE, Weekly_Sales FLOAT64> AS
  SELECT Store, Date, Weekly_Sales
  FROM `bigdata3844.RevData.walmart`
  WHERE Weekly_Sales > min_sales;
```

```
-- Using the Table Function
SELECT *
FROM `bigdata3844.RevData.filter_sales`(750000);
```

- **Real-World Scenario**: This function can be used by different departments to analyze stores that exceed certain sales thresholds.

4. SQL Stored Procedures

Stored Procedures in BigQuery allow you to execute multiple SQL statements in a procedural manner. This can include variable declarations, loops, conditionals, and more, making them ideal for complex logic that involves multiple steps.

Example: A Stored Procedure for Sales Summary Reporting

```
CREATE OR REPLACE PROCEDURE
`bigdata3844.RevData.procedures.generate_sales_report`(start_date DATE,
end_date DATE)
BEGIN
  DECLARE total_sales FLOAT64 DEFAULT 0;
  DECLARE store_count INT64 DEFAULT 0;

  -- Calculate total sales within the date range
  SET total_sales = (
    SELECT SUM(Weekly_Sales)
    FROM `bigdata3844.RevData.walmart`
    WHERE Date BETWEEN start_date AND end_date
  );

  -- Count the number of unique stores within the date range
  SET store_count = (
    SELECT COUNT(DISTINCT Store)
    FROM `bigdata3844.RevData.walmart`
    WHERE Date BETWEEN start_date AND end_date
  );

  -- Output the results
  SELECT start_date AS Start_Date, end_date AS End_Date, total_sales AS
Total_Sales, store_count AS Store_Count;
END;
```

```
CALL `bigdata3844.RevData.p_generate_sales_report`('2010-01-01', '2020-12-
31');
```

- **Real-World Scenario**: This stored procedure could be scheduled to run weekly, providing a summary report to business analysts.

5. Practical Considerations

- **Performance**: UDFs and Table Functions should be used judiciously, as they can impact query performance, especially when processing large datasets.
- **Security**: Ensure that UDFs and Stored Procedures do not expose sensitive data inadvertently, especially when shared across teams.
- **Testing**: Regularly test and review routines to ensure they remain performant and correct as data structures evolve.

6. Conclusion

BigQuery Routines are a powerful feature that provides flexibility, reusability, and scalability in managing complex data workflows. By mastering these, you can significantly optimize your BigQuery operations, ensuring efficient and maintainable code that meets real-world business needs.