BigQuery Setup Guide for E-commerce Data Generator

Overview

This guide will help you set up BigQuery integration for your synthetic e-commerce data generator, migrating from CSV files to direct BigQuery ingestion.

Prerequisites

- Google Cloud Platform account
- Basic familiarity with BigQuery
- Python 3.7+ environment

Step 1: Google Cloud Setup

1.1 Create/Select GCP Project

```bash
# Install gcloud CLI if not already installed
# Visit: https://cloud.google.com/sdk/docs/install

# Login to Google Cloud
gcloud auth login

# Create new project (or use existing)
gcloud projects create your-ecommerce-project --name="E-commerce Analytics"

# Set as default project
gcloud config set project your-ecommerce-project
```

1.2 Enable Required APIs

```bash
# Enable BigQuery API
gcloud services enable bigquery.googleapis.com

# Enable BigQuery Data Transfer API (optional, for advanced features)
gcloud services enable bigquerydatatransfer.googleapis.com
```

1.3 Set up Billing

- Go to Google Cloud Console
- Navigate to Billing → Link a billing account
- BigQuery offers generous free tier: 1TB queries/month, 10GB storage free

Step 2: Authentication Setup

Option A: Service Account (Recommended for Production)

1. **Create Service Account:**

```bash
# Create service account
gcloud iam service-accounts create ecommerce-data-loader \
    --description="Service account for e-commerce data loading" \
    --display-name="E-commerce Data Loader"

# Grant BigQuery permissions
gcloud projects add-iam-policy-binding your-ecommerce-project \
    --member="serviceAccount:ecommerce-data-loader@your-ecommerce-project.iam.gserviceaccount.com" \
    --role="roles/bigquery.admin"
```

2. **Download Key File:**

```bash
# Generate and download key file
gcloud iam service-accounts keys create ~/ecommerce-service-account-key.json \
    --iam-account=ecommerce-data-loader@your-ecommerce-project.iam.gserviceaccount.com
```

3. **Set Environment Variable:**

```bash
export GOOGLE_APPLICATION_CREDENTIALS="~/ecommerce-service-account-key.json"
```

Option B: Application Default Credentials (Easy for Development)

```bash
```

```
# Authenticate with your user account
gcloud auth application-default login

# Set default project
gcloud config set project your-ecommerce-project
```

Step 3: Python Environment Setup

3.1 Install Dependencies

bash

```bash
# Install required packages
pip install google-cloud-bigquery pandas numpy faker python-dateutil

# Or use requirements.txt
pip install -r requirements.txt
```

3.2 Requirements.txt

text

```text
google-cloud-bigquery>=3.10.0
pandas>=1.5.0
numpy>=1.21.0
faker>=18.0.0
python-dateutil>=2.8.0
```

Step 4: Configuration

4.1 Update Script Configuration

Edit the `CONFIG` dictionary in your Python script:

python

```
CONFIG = {
    'num_customers': 25000,
    'num_products': 2500,
    'num_promotions': 75,
    'start_date': datetime(2022, 1, 1),
    'end_date': datetime(2024, 7, 31),
    'data_period_days': 912,

    # BigQuery Configuration - UPDATE THESE
    'project_id': 'your-ecommerce-project',  # Your GCP project ID
    'dataset_id': 'ecommerce_analytics',     # Dataset name
    'location': 'US',                        # BigQuery location
    'service_account_path': '~/ecommerce-service-account-key.json',  # If using service account
    'batch_size': 10000,                     # Records per batch
}
```

4.2 Verify Configuration

```python
# Test BigQuery connection
from google.cloud import bigquery

client = bigquery.Client(project='your-ecommerce-project')
print(f"BigQuery client created for project: {client.project}")

# List datasets (should be empty initially)
datasets = list(client.list_datasets())
print(f"Datasets in project: {[d.dataset_id for d in datasets]}")
```

Step 5: Create BigQuery Schema

5.1 Run Schema Creation Script

Execute the SQL schema creation script in BigQuery Console or via command line:

```bash
# Using bq command line tool
bq query < bigquery_schema.sql

# Or run directly in BigQuery Console
# Copy and paste the SQL from the schema artifact
```

5.2 Verify Schema Creation

```sql
-- Check if dataset was created
SELECT
  schema_name,
  location
FROM `your-ecommerce-project.INFORMATION_SCHEMA.SCHEMATA`
WHERE schema_name = 'ecommerce_analytics';

-- List tables in dataset
SELECT
  table_name,
  table_type,
  creation_time
FROM `your-ecommerce-project.ecommerce_analytics.INFORMATION_SCHEMA.TABLES`;
```

Step 6: Run Data Generation

6.1 Execute the Script

```bash
python bigquery_synthetic_data_generator.py
```

6.2 Monitor Progress

The script will show progress indicators:

```
=== E-commerce Synthetic Data Generation for BigQuery ===
✓ BigQuery connection established
✓ Dataset ecommerce_analytics already exists
Generating customers...
✓ Created 25,000 customers
Generating products...
✓ Created 2,500 products
Generating promotions...
✓ Created 75 promotions

=== Loading Foundation Tables to BigQuery ===
✓ Loaded 25,000 records to customers
✓ Loaded 2,500 records to products
✓ Loaded 75 records to promotions
```

Step 7: Verification and Testing

7.1 Data Quality Checks

```
sql
```

```
-- Check customer distribution
SELECT
  customer_segment,
  COUNT(*) as count,
  ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER(), 2) as percentage
FROM `your-ecommerce-project.ecommerce_analytics.customers`
GROUP BY customer_segment
ORDER BY count DESC;

-- Check date ranges
SELECT
  MIN(registration_date) as earliest_registration,
  MAX(registration_date) as latest_registration,
  COUNT(DISTINCT customer_id) as unique_customers
FROM `your-ecommerce-project.ecommerce_analytics.customers`;

-- Product category distribution
SELECT
  category,
  COUNT(*) as product_count,
  ROUND(AVG(base_price), 2) as avg_price
FROM `your-ecommerce-project.ecommerce_analytics.products`
GROUP BY category
ORDER BY product_count DESC;
```

7.2 Cross-table Relationships

```sql
-- Verify referential integrity
SELECT
  'orders_to_customers' as check_type,
  COUNT(*) as total_orders,
  COUNT(DISTINCT c.customer_id) as customers_with_orders
FROM `your-ecommerce-project.ecommerce_analytics.orders` o
LEFT JOIN `your-ecommerce-project.ecommerce_analytics.customers` c
  ON o.customer_id = c.customer_id;
```

Step 8: Cost Optimization

8.1 Partitioning and Clustering

The schema includes optimized partitioning:

- Date-based partitioning for time-series data
- Clustering on frequently filtered columns
- Appropriate data types for storage efficiency

8.2 Query Optimization

```sql
-- Use partition pruning
SELECT *
FROM `your-ecommerce-project.ecommerce_analytics.orders`
WHERE order_date BETWEEN '2024-01-01' AND '2024-01-31'  -- Uses partition pruning
  AND customer_segment = 'high_value';  -- Uses clustering

-- Avoid SELECT *
SELECT customer_id, order_date, order_value
FROM `your-ecommerce-project.ecommerce_analytics.orders`
WHERE order_date >= '2024-01-01';
```

8.3 Cost Monitoring

```sql
-- Check dataset size
SELECT
  table_name,
  ROUND(size_bytes / 1024 / 1024, 2) as size_mb,
  row_count
FROM `your-ecommerce-project.ecommerce_analytics.__TABLES__`
ORDER BY size_bytes DESC;
```

Troubleshooting

Common Issues

1. **Authentication Errors**

   ```
   Error: google.auth.exceptions.DefaultCredentialsError
   ```

   - Solution: Check `gcloud auth application-default login` or service account setup

2. **Permission Denied**

   ```
   Error: 403 Access Denied
   ```

- Solution: Ensure BigQuery Admin role assigned to user/service account

3. **Project Not Found**

   ```
   Error: 400 Project not found
   ```

   - Solution: Verify project ID in CONFIG matches actual GCP project

4. **Quota Exceeded**

   ```
   Error: 403 Quota exceeded
   ```

   - Solution: Check BigQuery quotas in GCP console, consider reducing batch size

5. **Schema Mismatch**

   ```
   Error: Schema mismatch
   ```

   - Solution: Drop and recreate tables, or adjust data types in preparation function

Performance Tips

1. **Large Datasets**: Increase `batch_size` for better throughput
2. **Memory Issues**: Reduce `num_customers` or process in smaller chunks
3. **Network Timeouts**: Add retry logic or reduce batch sizes
4. **Schema Evolution**: Use `WRITE_APPEND` instead of `WRITE_TRUNCATE` for incremental loads

Support Resources

- BigQuery Documentation
- Python Client Documentation
- BigQuery Pricing
- GCP Support

Next Steps

After successful setup:

1. **Explore Data**: Use BigQuery Console to explore your generated data
2. **Build Dashboards**: Connect to Looker Studio, Tableau, or other BI tools
3. **ML Integration**: Use BigQuery ML for churn prediction models
4. **Automation**: Set up scheduled data generation using Cloud Functions
5. **Data Pipeline**: Integrate with Cloud Composer/Airflow for production workflows