Low-Level Technical Design Document Project: GreenQuery - Sustainable BigQuery Optimization **Date:** July 22, 2025 **Team:** Lloyds Technology Centre

---

1. Objective

To build a cost-effective, performance-optimized, and sustainable BigQuery monitoring and optimization platform called **GreenQuery** that:

- Detects and recommends fixes for inefficient queries.
- Estimates cost and CO2 emissions.
- Provides actionable dashboards and alerts for data teams.

---

2. Tech Stack

- **Cloud Platform:** Google Cloud Platform (GCP)
- **Primary Services:** BigQuery, Cloud Functions, Cloud Scheduler, Pub/Sub, Cloud Logging
- **Monitoring & Reporting:** Looker Studio
- **Programming Languages:** Python, SQL
- **ML/Forecasting:** BigQuery ML
- **CI/CD Integration:** GitHub Actions

---

3. Project Structure

```
greenquery/
├── config/
│   ├── dry_run_config.json
│   └── alert_rules.yaml
├── data/
│   ├── raw_logs/
│   └── processed/
├── functions/
│   ├── dry_run_estimator.py
│   ├── log_parser.py
│   ├── notifier.py
│   └── optimization_advisor.py
├── ml/
│   ├── query_classifier.sql
│   └── forecasting_model.sql
├── reports/
│   └── dashboard_templates/
├── scripts/
│   └── setup_bigquery_views.sql
```

```
├── tests/
│   └── test_dry_run_estimator.py
└── main.py
```

---

4. Component Breakdown

A. Log Collection

- **Source:** BigQuery audit logs
- **Method:** Sink logs to a central GCP logging bucket and export to BigQuery
- **Storage:** `greenquery_logs.raw_logs`

B. Log Parsing

- **Script:** `log_parser.py`
- **Function:** Extract relevant fields (user, query, project, data processed, etc.) and store in `processed_logs` table.

C. Dry Run Estimation

- **Script:** `dry_run_estimator.py`
- **Function:** Runs BigQuery jobs in `dry_run=True` mode and captures estimated cost and data scanned.
- **Trigger:** Cloud Function triggered via Pub/Sub or scheduler

D. Query Classification (ML)

- **Tool:** BigQuery ML
- **Script:** `ml/query_classifier.sql`
- **Function:** Classify queries as efficient or inefficient
- **Input:** `processed_logs`

E. Forecasting Model (ML)

- **Tool:** BigQuery ML ARIMA_PLUS
- **Script:** `ml/forecasting_model.sql`
- **Function:** Forecast future cost and emissions based on query patterns

F. Optimization Advisory

- **Script:** `optimization_advisor.py`
- **Function:** Provide tips (e.g., add filter, partition, cluster, avoid SELECT *)

G. Alerting and Notifications

- **Script:** `notifier.py`

- **Trigger:** Cost/CO2 exceeds threshold (defined in `alert_rules.yaml`)
- **Channels:** Slack, Email

H. Dashboarding

- **Tool:** Looker Studio
- **Source Tables:** `processed_logs`, `classification_results`, `forecast_results`
- **Views:** Persona-based (Engineers, Analysts, Executives)

---

5. Data Flow

1. **BigQuery Logs → Log Sink → BQ Dataset (raw)**
2. **Raw Logs → Cloud Function (Parser) → Processed Logs Table**
3. **Dry Run Estimator → Estimated Cost/Emissions Table**
4. **ML Classifier + Forecasting → Results Tables**
5. **Advisor → Optimization Suggestions Table**
6. **Notifier → Alerts to Slack/Email**
7. **Looker Studio → Visual Dashboards**

---

6. Cost Optimization Measures

- Use **on-demand Cloud Functions** instead of long-running services
- Store logs in **compressed formats**
- Minimize storage via **partitioned tables**
- Use **caching and dry run** for forecasting without actual execution
- Prefer **scheduled batch** jobs over real-time where latency isn't critical

---

7. Future Enhancements

- Gemini-powered SQL rewrite suggestions
- CI/CD query budget guardrails
- Geo-based carbon estimation models
- Integration with enterprise ITSM (ServiceNow)

---

8. Security & IAM

- Use least-privilege roles for Cloud Functions
- Logs anonymized (PII masked) in `processed_logs`
- IAM scoped by team (Eng/Analyst/Exec)

---

9. Deployment

- **Setup Script:** `scripts/setup_bigquery_views.sql`
- **CI/CD:** GitHub Actions for test + deploy
- **Monitoring:** GCP Ops Suite (Logging + Monitoring)

---

10. Testing Strategy

- **Unit Tests:** For each Cloud Function in `/tests`
- **Integration Tests:** End-to-end dry run + optimization flow
- **Load Tests:** Simulate logs using synthetic queries

---

11. Deliverables

- Source code repo (GitHub)
- GCP deployment templates (Terraform or scripts)
- Documentation (README, architecture, runbooks)
- Dashboards & sample alerts