GA4 Anomaly Detection Solution

Contributors(Group-5)

Dhananjay Kanjariya
Ronit Rajput
Aarya Samaiya
Vishnu Nair

Mentors

Ravi Pathak
Sarjak Patel
Khilav Joshi
Dharmik Raval
Ruchika Parikh

31st January 2026

1 Introduction

Google Analytics 4 (GA4) produces high-volume event-level data that must be continuously monitored to ensure analytical reliability and business accuracy. This project implements a fully automated GA4 anomaly detection solution using synthetic data, BigQuery Scheduled Queries, and BigQuery ML. The pipeline detects statistical anomalies, maps them to business severity, and delivers actionable alerts through an end-to-end, production-grade architecture. A key design principle of the system is strict separation of concerns: statistical detection, business interpretation, and alert delivery are fully decoupled. This enables explainability, auditability, and safe evolution of business rules while closely mirroring real-world enterprise analytics monitoring platforms.

2 Resources(with hyperlinks)

2.1 Datasets

- **GA4Sample_live** — Synthetic GA4-style raw event dataset used to simulate daily GA4 BigQuery exports with seasonality and injected anomalies.
- **analytics_live** — Analytical dataset containing aggregated metrics, forecasting models, anomaly outputs, severity scores, and alert decisions.

2.2 Scheduled Queries

- **daily_data_generator** — Generates daily synthetic GA4 events
- **anomaly_injector** — Injects controlled anomalies for validation
- **loading_to_daily_metric** — Aggregates events into daily metrics
- **processing_gap_filling** — Ensures metric continuity
- **calculating_statistical_metric** — Forecasting and anomaly detection
- **severity_business_logic** — Severity and impact classification
- **alert_decision** — Alert eligibility and suppression logic
- **ARIMA_retrain_30_days** — Periodic model retraining

2.3 Alert Output

- **Email Alerts** — Automated, professionally formatted anomaly notifications delivered via Google Apps Script.

3 System Architecture

The GA4 anomaly detection platform is implemented as a **fully serverless, batch-oriented analytics system** built entirely on Google BigQuery and BigQuery ML. All computation, modeling, and decision-making logic is executed using BigQuery Scheduled Queries, while alert delivery is handled separately via Google Apps Script.

The architecture is designed to reflect production-grade analytics monitoring systems, emphasizing determinism, auditability, and strict separation of responsibilities across pipeline stages.

3.1 High-Level Architectural Overview

At a high level, the system transforms raw GA4-style events into business-critical alerts through a sequence of isolated, idempotent processing layers:

1. Synthetic GA4 event generation
2. Controlled anomaly injection (optional)
3. Daily metric aggregation
4. Metric gap filling
5. Time-series forecasting and anomaly detection
6. Severity and business impact classification
7. Alert eligibility and suppression decisioning
8. Automated alert delivery

Each stage produces a materialized output that serves as the sole input to the next stage. No stage directly depends on upstream raw data once its input table has been created.

3.2 Separation of Analytical and Operational Layers

A core architectural principle of the system is the strict separation between:

- **Analytical decision-making** (BigQuery)
- **Operational notification delivery** (Apps Script)

All statistical modeling, severity classification, business impact assessment, and alert eligibility logic is implemented declaratively in BigQuery. The delivery layer consumes only finalized alert records and performs no independent evaluation.

This design ensures that:

- Business logic is centralized and auditable
- Alert behavior can be validated via SQL alone
- Delivery mechanisms can be replaced without affecting analytics

3.3 Dataset and Layering Strategy

To mirror real-world production environments, the pipeline is organized into two logical datasets:

- **GA4SampleData_live** — Raw, synthetic GA4-style event tables
- **analytics_live** — Derived metrics, models, and decisions

This separation enforces a clean boundary between data ingestion and analytics processing, preventing accidental coupling between raw inputs and downstream decisions.

3.4 Materialized Dependency Chain

The system is structured as a linear dependency graph in which each table has a single responsibility:

```
events_YYYYMMDD
      ↓
ga4_event_metrics_daily
      ↓
ga4_event_metrics_daily_filled
      ↓
ARIMA_PLUS models (per metric)
      ↓
ga4_anomaly_enriched_all_events
      ↓
ga4_anomaly_scored_events
      ↓
ga4_anomaly_alert_decisions
      ↓
email_payload_view
      ↓
Apps Script (email delivery)
```

Each arrow represents a strict data dependency. No stage reads from tables further upstream than its immediate predecessor.

3.5 Execution and Orchestration Model

All analytical stages are orchestrated using BigQuery Scheduled Queries executed in UTC. However, all business logic—including date resolution, aggregation windows, and anomaly evaluation—operates in the `Asia/Kolkata` timezone.

The system consistently applies the following rule:

> *Each scheduled job processes the previous business day (IST).*

This ensures that:

- Partial-day data is never evaluated
- All pipeline stages agree on the same processing date
- Downstream joins remain deterministic

3.6 Idempotency and Determinism

Every scheduled query follows a strict idempotency pattern:

1. Resolve the processing date
2. Delete existing records for that date
3. Insert freshly computed results

This guarantees that:

- Any day can be safely recomputed
- Failures do not propagate across dates
- Backfills and replays require no code changes

3.7 Model Isolation and Reusability

Time-series models are trained independently for each metric. This avoids cross-metric interference and allows:

- Metric-specific sensitivity tuning
- Independent retraining schedules
- Clear attribution of anomalies to individual signals

Forecasting models are treated as read-only dependencies during daily anomaly detection, ensuring stability and reproducibility of results.

3.8 Why This Architecture Is Production-Grade

The architectural choices made in this system reflect best practices used in enterprise analytics monitoring platforms:

- Stateless, batch-oriented computation
- Explicit materialization at every stage
- Deterministic, timezone-safe execution
- Clear separation between analytics and alerting

As a result, the system remains robust under operational failures, scales with data volume, and supports future extensions such as additional metrics, root-cause analysis, or alternative alert channels without architectural refactoring.

4 Data Sources and Schemas

The GA4 anomaly detection pipeline is built on a layered data model inspired by modern analytics engineering practices. Each dataset and table serves a single, well-defined responsibility, enabling clarity, scalability, and deterministic downstream processing.

A fundamental design principle is that **every downstream decision is traceable to a concrete, materialized upstream dataset**. No implicit transformations or hidden state are permitted.

4.1 Dataset Overview

The system uses two primary datasets:

- **GA4SampleData_live**: Raw, synthetic GA4-style event data
- **analytics_live**: All derived metrics, models, anomaly outputs, severity classifications, and alert decisions

This separation mirrors production analytics architectures, where raw ingestion and analytical processing are strictly decoupled.

—

4.2 Raw Event Data: GA4SampleData_live

4.2.1 Purpose

The `GA4SampleData_live` dataset simulates the native Google Analytics 4 BigQuery export. It represents the immutable source of truth for all downstream computations.

Synthetic events are generated to closely match GA4 semantics, cardinality, and schema structure, allowing the pipeline to be migrated to real GA4 exports without architectural changes.

4.2.2 Table Naming Convention

`events_YYYYMMDD`

Each table contains all GA4 events for a single business day.

4.2.3 Expanded Core Schema: events_YYYYMMDD

Field Name	Type	Mode	Description
event_name	STRING	NULLABLE	GA4 event identifier (e.g., page_view, purchase)
event_params	RECORD	REPEATED	Key-value parameters associated with the event
event_previous_timestamp	INTEGER	NULLABLE	Timestamp of previous related event
event_value_in_usd	FLOAT	NULLABLE	Monetary value normalized to USD (if applicable)
event_bundle_sequence_id	INTEGER	NULLABLE	Sequence ID for bundled events
event_server_timestamp_offset	INTEGER	NULLABLE	Server-side timestamp offset
user_id	STRING	NULLABLE	Logged-in user identifier
user_pseudo_id	STRING	NULLABLE	Anonymous GA4 user identifier
privacy_info	RECORD	NULLABLE	Consent and privacy metadata
user_properties	RECORD	REPEATED	User-scoped attributes
user_first_touch_timestamp	INTEGER	NULLABLE	First interaction timestamp
user_ltv	RECORD	NULLABLE	Lifetime value metadata
device	RECORD	NULLABLE	Device information (OS, browser, category)
geo	RECORD	NULLABLE	Geographic attributes (country, region, city)
app_info	RECORD	NULLABLE	App-specific metadata
traffic_source	RECORD	NULLABLE	Acquisition source information
stream_id	INTEGER	NULLABLE	Data stream identifier
platform	STRING	NULLABLE	Platform type (WEB / ANDROID / IOS)
event_dimensions	RECORD	NULLABLE	Custom event dimensions
ecommerce	RECORD	NULLABLE	Ecommerce revenue and transaction data
items	RECORD	REPEATED	Item-level ecommerce details
event_timestamp	INTEGER	NULLABLE	Event timestamp in microseconds
event_date	STRING	NULLABLE	Event date in YYYYMMDD format

4.3 Derived Analytics Layer: analytics_live

The `analytics_live` dataset contains all transformed, modeled, and decision-oriented tables. Unlike raw data, tables in this dataset are **mutable and recomputable**, following strict idempotency guarantees.

4.3.1 Table: ga4_event_metrics_daily

Purpose Aggregates raw GA4 events into daily metric values suitable for time-series modeling and anomaly detection.

Schema

Field Name	Type	Mode	Description
event_date	DATE	NULLABLE	IST-aligned business processing date
event_name	STRING	NULLABLE	Metric name derived from GA4 event
event_value	FLOAT	NULLABLE	Aggregated daily metric value

—

4.3.2 Table: ga4_event_metrics_daily_filled

Purpose Ensures continuity of all metrics by filling missing metric-date combinations with zero values.

Schema

event_date	DATE	NULLABLE	Processing date
event_name	STRING	NULLABLE	Metric identifier
event_value	FLOAT	NULLABLE	Gap-filled metric value

—

4.3.3 Table: ga4_anomaly_enriched_all_events

Purpose Stores pure statistical anomaly signals enriched with forecasts, confidence intervals, and probabilities.

Schema

event_date	DATE	NULLABLE	Processing date
event_name	STRING	NULLABLE	Metric name
actual_value	FLOAT	NULLABLE	Observed metric value
expected_value	FLOAT	NULLABLE	Forecasted value
lower_bound	FLOAT	NULLABLE	Prediction interval lower bound
upper_bound	FLOAT	NULLABLE	Prediction interval upper bound
deviation_pct	FLOAT	NULLABLE	Relative deviation from forecast
z_score	FLOAT	NULLABLE	Standardized deviation score
bound_distance	FLOAT	NULLABLE	Distance beyond confidence interval
is_outside_bounds	BOOLEAN	NULLABLE	Interval breach flag
is_anomaly	BOOLEAN	NULLABLE	ML anomaly classification
anomaly_probability	FLOAT	NULLABLE	Anomaly confidence score
computation_timestamp	TIMESTAMP	NULLABLE	Record creation timestamp

—

4.3.4 Table: ga4_anomaly_scored_events

Purpose Translates statistical anomalies into business-aware severity and impact classifications, and attaches root-cause context.

Schema

event_date	DATE	NULLABLE	Processing date
event_name	STRING	NULLABLE	Metric name
actual_value	FLOAT	NULLABLE	Observed value
expected_value	FLOAT	NULLABLE	Forecasted value
lower_bound	FLOAT	NULLABLE	Lower prediction bound
upper_bound	FLOAT	NULLABLE	Upper prediction bound
deviation_pct	FLOAT	NULLABLE	Relative deviation
anomaly_probability	FLOAT	NULLABLE	ML confidence score
z_score	FLOAT	NULLABLE	Standardized deviation
bound_distance	FLOAT	NULLABLE	Distance beyond bounds
is_outside_bounds	BOOLEAN	NULLABLE	Interval breach indicator
is_anomaly	BOOLEAN	NULLABLE	Anomaly flag
severity_level	STRING	NULLABLE	Severity classification
business_impact	STRING	NULLABLE	Business impact level
suspected_root_cause	STRING	NULLABLE	Human-readable root cause
recommended_actions	STRING	REPEATED	Suggested remediation steps

4.3.5 Table: ga4_anomaly_alert_decisions

Purpose Determines alert eligibility, suppression, repetition, and priority.

Schema

event_date	DATE	NULLABLE	Processing date
event_name	STRING	NULLABLE	Metric identifier
severity_level	STRING	NULLABLE	Severity level
business_impact	STRING	NULLABLE	Business impact classification
alert_eligible	BOOLEAN	NULLABLE	Alert eligibility flag
repeated_alert	BOOLEAN	NULLABLE	Repetition indicator
suppressed	BOOLEAN	NULLABLE	Suppression decision
alert_priority	STRING	NULLABLE	Assigned alert priority
decision_timestamp	TIMESTAMP	NULLABLE	Decision creation timestamp

4.4 Schema Design Rationale

Across all datasets, schema design adheres to three principles:

- **Explicitness**: All derived values are materialized
- **Traceability**: Every decision is traceable to upstream data
- **Extensibility**: New metrics or models can be added without schema redesign

This layered schema strategy enables independent validation of each pipeline stage and supports long-term evolution of the anomaly detection platform.

5 Synthetic GA4 Data Generation

5.1 Motivation

At the time of system development, a live Google Analytics 4 (GA4) production data connection was not available. Additionally, time-series forecasting models such as ARIMA_PLUS require long, continuous historical data (typically 9–12 months) to accurately learn trends, seasonality, and variance characteristics.

To address these constraints, a synthetic GA4 data generation framework was designed with the following objectives:

- Simulate realistic GA4 event distributions
- Preserve schema compatibility with GA4 BigQuery exports
- Introduce natural seasonality and holiday effects
- Inject controlled, non-deterministic anomalies
- Enable deterministic daily backfilling and replay

This design ensures that downstream aggregation, forecasting, and anomaly detection logic behaves identically to a real production GA4 pipeline.

5.2 Synthetic Event Coverage

The synthetic generator produces a comprehensive set of GA4 events representing the complete e-commerce user funnel:

- page_view
- session_start
- user_engagement
- scroll
- view_item
- add_to_cart
- begin_checkout
- add_payment_info
- purchase

Each event is generated with an independent probability that reflects realistic drop-off behavior observed in real-world digital commerce platforms.

5.3 User Simulation Strategy

Synthetic users are sampled from a persistent user pool stored in `analytics_live.synthetic_users`. For each day, a random subset of users is selected to simulate organic churn, repeat visits, and session continuity.

Design Rationale

- Prevents unrealistic regeneration of users every day
- Enables stable engagement trends across time
- Mimics returning and inactive user behavior

This approach aligns with GA4's pseudonymous user tracking model.

5.4 Daily Event Volume Modeling

For each processing date, event generation follows a probabilistic model:

$$P(event) = base\_probability \times seasonal\_multiplier \times holiday\_multiplier \quad (1)$$

This formulation allows independent control of baseline behavior, weekly seasonality, and exceptional calendar effects.

5.4.1 Base Event Probabilities

Event	Base Probability
page_view	0.30
session_start	0.10
user_engagement	0.10
scroll	0.20
view_item	0.15
add_to_cart	0.03
begin_checkout	0.02
add_payment_info	0.015
purchase	0.003

These probabilities approximate real conversion funnels and ensure realistic metric distributions.

5.5 Seasonality Modeling

Weekly seasonality is introduced using weekday-based multipliers:

- Weekdays (Monday–Friday): multiplier = 1.00
- Weekends (Saturday–Sunday): multiplier = 0.85

This reflects lower commercial engagement typically observed on weekends.

5.6 Holiday Effects

Explicit holiday multipliers are applied to prevent legitimate demand spikes from being incorrectly classified as anomalies.

5.6.1 Supported Holidays

- Diwali (Oct 28 – Nov 2): 1.8x
- Christmas (Dec 25): 1.5x

Rationale Encoding holiday behavior enables ARIMA_PLUS models to correctly decompose seasonal components rather than treating expected surges as anomalous behavior.

5.7 Revenue Modeling

Purchase events include a nested `ecommerce` structure consistent with GA4 exports:

- Revenue range: INR 500 – INR 4500
- Currency fixed as INR
- Revenue randomized per transaction

This enables revenue-based anomaly detection instead of relying solely on event counts.

5.8 Anomaly Injection Strategy

Anomalies are injected through a separate scheduled process that runs **after** clean data generation.

5.8.1 Trigger Probability

Each day has a 7% probability of being designated an anomaly day.

5.8.2 Anomaly Types

- **Negative Traffic Anomalies**:
  - Random deletion of 40–50% of page_view, session_start, and user_engagement events
- **Positive Revenue Spikes**:
  - Duplication of 50–60% of purchase events

5.9 Scheduling and Idempotency

Synthetic data generation is executed once per day using BigQuery Scheduled Queries.

- Processing date = previous calendar day (IST-safe)
- Tables created using `CREATE OR REPLACE`
- Anomaly injection is conditional and reversible

This guarantees reproducibility, deterministic backfills, and safe reprocessing.

5.10 Benefits of Synthetic Data Design

The synthetic data framework provides:

- Production-realistic GA4 behavior
- Long historical coverage for forecasting models
- Controlled anomaly labeling for validation
- Seamless replacement with live GA4 exports

As a result, all downstream analytics, forecasting, and anomaly detection logic can be migrated to real GA4 data without architectural changes.

6 Daily Metric Aggregation

6.1 Purpose and Motivation

Raw GA4 event data is highly granular and not directly suitable for time-series modeling or anomaly detection. Forecasting models such as ARIMA_PLUS require a single numeric value per metric per day, rather than millions of individual event records.

The objective of the daily metric aggregation stage is therefore to:

- Transform event-level GA4 data into daily business metrics
- Standardize metrics into a uniform time-series format
- Separate engagement metrics from revenue metrics
- Create a deterministic, idempotent aggregation layer

This stage forms the foundation for all downstream forecasting and anomaly detection logic.

6.2 Input Data

The aggregation process consumes synthetic GA4 event tables stored in the `GA4SampleData_live` dataset.

- Source tables follow the naming convention:

  ```
  events_YYYYMMDD
  ```

- Each table represents a single calendar day of GA4 events
- Event schema closely mirrors official GA4 BigQuery exports

6.3 Output Table

Aggregated metrics are written to the table:

`analytics_live.ga4_event_metrics_daily`

Each row represents a single metric value for a given date.

6.4 Metric Definitions

Two types of metrics are produced:

6.4.1 Count-Based Metrics

The following engagement metrics are computed using event counts:

- page_view
- session_start
- user_engagement
- add_to_cart
- add_payment_info

Aggregation Rule

```
metric_value = COUNT(*)
```

6.4.2 Revenue-Based Metrics

- purchase

Aggregation Rule

```
metric_value = SUM(ecommerce.purchase_revenue)
```

Revenue aggregation uses the nested GA4 `ecommerce` structure, ensuring schema consistency with real GA4 exports.

6.5 Processing Date Resolution

To support both live execution and historical replay, the processing date is resolved dynamically using a pipeline control table:

- Live mode processes the previous calendar day
- Replay mode processes a manually specified date

This logic enables safe backfilling without modifying SQL code.

6.6 Idempotency Strategy

Before inserting new data, existing rows for the target date are deleted. This ensures:

- Safe re-execution of scheduled queries
- No duplicate rows
- Deterministic outputs

6.7 Design Rationale

6.7.1 Why Separate Count and Revenue Metrics

GA4 events represent fundamentally different measurement types:

- Engagement events measure frequency
- Purchase events measure monetary value

Separating aggregation logic ensures numerical correctness and avoids misinterpretation of metrics.

6.7.2 Why Use EXECUTE IMMEDIATE

Daily GA4 event tables are date-suffixed and cannot be referenced statically. Dynamic SQL enables:

- Clean daily partitioning
- Automatic table resolution
- Scalable daily execution

6.7.3 Why Use COALESCE for Revenue

On days with zero purchases, `SUM()` returns NULL. Replacing NULL with zero ensures:

- Continuous time series
- Compatibility with forecasting models
- Explicit representation of zero-revenue days

6.8 Guarantees Provided by This Stage

The daily aggregation layer guarantees that:

- Each metric has at most one value per day
- Metrics are numerically consistent
- Downstream stages receive clean, standardized inputs

This table serves as the canonical source for all forecasting and anomaly detection models in the system.

7 Gap Filling Strategy

7.1 Motivation

Time-series forecasting models such as ARIMA_PLUS require a **continuous, regularly spaced time series**. In real-world analytics pipelines, however, daily metrics may be missing for certain dates due to:

- Zero user activity for a metric on a given day
- Partial upstream failures or ingestion delays
- Conditional synthetic anomaly injection

If these gaps are not handled explicitly, forecasting models may:

- Fail to train or forecast
- Infer incorrect seasonality
- Produce unstable or misleading anomaly scores

To prevent these issues, a dedicated gap-filling stage is introduced to ensure that **every metric exists for every processing date**, even if its value is zero.

7.2 Design Goals

The gap-filling layer is designed to:

- Guarantee metric continuity across all dates
- Preserve original observed values without modification
- Explicitly represent missing metrics as zero-valued observations
- Remain idempotent and replay-safe

This design ensures that downstream models operate on a complete and stable time series.

7.3 Input and Output Tables

7.3.1 Input Table

`analytics_live.ga4_event_metrics_daily`

Contains only metrics that were observed on a given day.

7.3.2 Output Table

`analytics_live.ga4_event_metrics_daily_filled`

Contains one row per metric per day, regardless of activity.

7.4 Metrics Covered

The gap-filling logic explicitly supports the following metrics:

- page_view
- session_start
- user_engagement
- add_to_cart
- add_payment_info
- purchase

This explicit enumeration avoids accidental metric drift and ensures schema stability over time.

7.5 Processing Date Resolution

As with other pipeline stages, the processing date is resolved dynamically to support both live execution and historical replay:

- Live mode processes the previous calendar day
- Replay mode processes a manually specified date

This allows any historical day to be recomputed without modifying SQL code.

7.6 Idempotency Strategy

Before inserting new records, existing rows for the target date are deleted. This ensures:

- Safe re-execution of scheduled queries
- No duplicate metric rows
- Deterministic outputs

7.7 Why LEFT JOIN with COALESCE

The gap-filling logic relies on two key SQL constructs:

7.7.1 LEFT JOIN

A LEFT JOIN ensures that all expected metrics are retained even if no corresponding row exists in the daily metrics table.

7.7.2 COALESCE

```
COALESCE(d.event_value, 0)
```

Explicitly converts missing values to zero, preserving numeric continuity without fabricating activity.

7.8 Why Zero is the Correct Default

Using zero as the default value for missing metrics is intentional:

- Zero represents the absence of activity, not missing data
- ARIMA_PLUS requires numeric continuity
- Forecast intervals remain interpretable
- Anomaly detection logic remains stable

Alternative approaches such as interpolation were avoided to prevent introducing artificial signal into the data.

7.9 Guarantees Provided by Gap Filling

After this stage, the system guarantees that:

- Every metric exists for every processing date
- No NULL values propagate into forecasting models
- Time series are contiguous and model-ready

This table serves as the **single source of truth** for all time-series modeling and anomaly detection stages.

7.10 Role in the Overall Pipeline

The gap-filled metrics table represents the final preprocessing step before machine learning is applied. By enforcing strict completeness guarantees, it allows forecasting and anomaly detection logic to remain simple, stable, and mathematically well-defined.

8 Time Series Modeling

8.1 Motivation

Daily engagement and conversion metrics derived from GA4 data exhibit strong temporal structure, including:

- Long-term trends (growth or decline in traffic)
- Weekly seasonality (weekday vs weekend behavior)
- Event-driven effects (holidays, promotions)
- Random noise and irregular variance

To accurately forecast expected behavior and detect anomalies, the system requires a model that can:

- Learn trend and seasonality automatically
- Provide statistically meaningful prediction intervals
- Scale across multiple independent metrics
- Operate fully inside BigQuery without external dependencies

BigQuery ML's `ARIMA_PLUS` model satisfies all of these requirements, making it well-suited for production-grade analytics pipelines.

8.2 Why ARIMA_PLUS

ARIMA_PLUS is an extension of classical ARIMA models that automatically:

- Selects optimal ARIMA parameters
- Decomposes time series into trend, seasonality, and residuals
- Handles missing values gracefully
- Produces confidence intervals for forecasts

Unlike black-box machine learning models, ARIMA_PLUS provides **interpretable statistical outputs**, which is critical for auditing anomaly decisions.

8.3 Per-Metric Modeling Strategy

Rather than training a single multivariate model, the pipeline trains **one independent ARIMA_PLUS model per metric**.

Metrics Modeled

- page_view
- session_start
- user_engagement
- add_to_cart
- add_payment_info
- purchase

Rationale

- Metrics have different scales and distributions
- Revenue behaves differently from engagement counts
- Independent models simplify diagnostics and retraining
- Prevents cross-metric interference

8.4 Training Data Source

All models are trained exclusively on the gap-filled metrics table:

`analytics_live.ga4_event_metrics_daily_filled`

This guarantees:

- Contiguous time series
- Absence of NULL values
- Stable seasonal decomposition

8.5 Model Configuration

Each ARIMA_PLUS model is created with the following configuration:

- `model_type = 'ARIMA_PLUS'`
- `time_series_timestamp_col = 'event_date'`
- `time_series_data_col = 'event_value'`
- `decompose_time_series = TRUE`

Why Decomposition is Enabled Time series decomposition allows the model to:

- Isolate weekly seasonality
- Separate trend from noise
- Prevent holidays from being misclassified as anomalies

8.6 Model Retraining Strategy

To ensure models remain adaptive to long-term changes, they are periodically retrained using a scheduled query.

Retraining Frequency

- Models are retrained every 30 days
- All historical data available at retrain time is used

This balances:

- Model freshness
- Computational cost
- Stability of learned seasonality

8.7 Why CREATE OR REPLACE

Using `CREATE OR REPLACE MODEL` ensures:

- Retraining is idempotent
- No model version drift
- Scheduled retrains are safe to re-run

This mirrors production MLOps best practices for batch model retraining.

8.8 Forecast Horizon Selection

All forecasts are generated with:

```
STRUCT(1 AS horizon)
```

Rationale

- Pipeline operates on daily cadence
- Anomaly detection is performed one day at a time
- Prevents compounding forecast uncertainty

8.9 Model Outputs Used Downstream

Each ARIMA_PLUS model produces:

- Forecasted value (`forecast_value`)
- Prediction interval lower bound
- Prediction interval upper bound
- Standard error

These outputs are consumed directly by the anomaly detection stage.

8.10 Benefits of This Modeling Approach

This time-series modeling strategy provides:

- Statistically interpretable forecasts
- Robust handling of seasonality and holidays
- Metric-specific anomaly sensitivity
- Fully serverless ML operations

The resulting models form the backbone of reliable, explainable anomaly detection in the final stage of the pipeline.

9 Anomaly Detection Logic

9.1 Objective

The objective of the anomaly detection stage is to identify statistically significant deviations in daily GA4-derived metrics relative to their expected behavior, while minimizing false positives caused by seasonality, holidays, or natural variance.

This stage transforms raw forecasts into actionable signals by combining:

- Time-series forecasting (expected behavior)
- Prediction intervals (statistical bounds)
- Probabilistic anomaly classification

The final output is a unified anomaly table suitable for alerting, monitoring dashboards, and post-incident analysis.

9.2 Why Forecast-Based Anomaly Detection

Simple threshold-based rules (e.g., fixed percentage drops) fail in real-world analytics systems because:

- Traffic naturally fluctuates by weekday
- Revenue variance increases with volume
- Seasonal trends shift over time

Forecast-based anomaly detection addresses these limitations by evaluating each observation relative to its **expected value for that specific day**.

9.3 Two-Layer Anomaly Decision Framework

The pipeline intentionally uses a **dual-signal approach**:

1. **Prediction interval breach** (deterministic)
2. **ML.DETECT_ANOMALIES classification** (probabilistic)

This ensures robustness, interpretability, and resilience to edge cases.

9.4 Layer 1: Prediction Interval Breach

ARIMA_PLUS produces a confidence interval around each forecast:

- Lower bound
- Upper bound

An observation is considered statistically suspicious if:

$$actual\_value < lower\_bound \quad \text{or} \quad actual\_value > upper\_bound \tag{2}$$

Why Prediction Intervals Matter

- Scale-aware (works for small and large metrics)
- Automatically adapts to variance
- Fully explainable to stakeholders

9.5 Layer 2: Probabilistic Anomaly Detection

BigQuery ML's `ML.DETECT_ANOMALIES` function provides:

- `is_anomaly` — boolean classification
- `anomaly_probability` — confidence score

This layer evaluates whether the observed deviation is unlikely under the model's learned distribution.

9.6 Metric-Specific Thresholds

Different GA4 metrics exhibit different volatility characteristics. Accordingly, anomaly sensitivity is tuned per metric.

9.6.1 Threshold Configuration

Metric	Threshold	Rationale
page_view	0.97	High volume, moderate variance
user_engagement	0.97	Correlated with traffic
add_to_cart	0.97	Funnel-sensitive metric
add_payment_info	0.97	Lower volume but stable
session_start	0.99	High natural variance
purchase	0.99	Revenue noise and spikes

Why Higher Thresholds for Revenue Revenue metrics are naturally spiky due to:

- Promotions
- Bulk purchases
- Seasonal campaigns

Using a higher threshold reduces false positives without missing true anomalies.

9.7 Derived Anomaly Features

Beyond binary anomaly flags, the pipeline computes additional diagnostic features to support investigation and alert prioritization.

9.7.1 Deviation Percentage

$$deviation\_pct = \frac{actual\_value - expected\_value}{expected\_value} \tag{3}$$

Provides scale-independent magnitude of deviation.

9.7.2 Z-Score Approximation

$$z\_score = \frac{actual\_value - expected\_value}{standard\_error} \tag{4}$$

Measures standardized distance from forecast.

9.7.3 Bound Distance

$$bound\_distance = \max(lower\_bound - actual\_value, actual\_value - upper\_bound, 0) \tag{5}$$

Quantifies how far an observation lies outside the confidence interval.

9.8 Daily Processing Strategy

Anomaly detection is executed once per day and always processes the **previous calendar day (IST)**.

Why Yesterday

- Ensures all events have landed
- Prevents partial-day false alarms
- Aligns with business reporting cycles

9.9 Idempotency Guarantees

Before inserting new anomaly records, the pipeline deletes any existing rows for the processing date:

```
DELETE FROM ga4_anomaly_enriched_all_events
WHERE event_date = run_date;
```

This ensures:

- Safe re-runs
- Deterministic outputs
- Clean backfills

9.10 Benefits of This Anomaly Detection Design

This anomaly detection framework provides:

- Statistically grounded alerts
- Explainable anomaly decisions
- Metric-specific sensitivity
- Robustness to seasonality and noise

The resulting enriched anomaly table serves as a reliable foundation for monitoring, alerting, and downstream analytics.

10 Scheduling and Timezone Handling

10.1 Why Scheduling Strategy Matters

In large-scale analytics pipelines, incorrect scheduling and timezone misalignment are among the most common sources of data inconsistency, partial aggregations, and false anomaly alerts.

Google Analytics 4 (GA4) event exports are inherently time-based, while BigQuery Scheduled Queries operate in UTC by default. Without explicit timezone handling, daily analytics pipelines may:

- Process incomplete days
- Split events across incorrect reporting dates
- Generate false anomalies due to partial data

To avoid these issues, the pipeline explicitly decouples **execution time** from **business date logic**.

10.2 Execution Time vs Business Date

Execution Time All BigQuery Scheduled Queries are configured and executed in UTC. This is a platform-level constraint and cannot be changed.

Business Date All analytics logic operates on dates derived in the `Asia/Kolkata` timezone, which represents the business reporting context for this system.

This distinction ensures that the pipeline behaves consistently regardless of infrastructure timezone.

10.3 The "Process Yesterday" Principle

Every stage in the pipeline processes the **previous calendar day** instead of the current day.

Definition

$$run\_date = \text{CURRENT\_DATE(Asia/Kolkata)} - 1 \tag{6}$$

Rationale

- Guarantees all events for the day have landed
- Prevents partial-day aggregations
- Aligns with end-of-day reporting cycles

This design is critical for anomaly detection, where even small partial-day gaps can be misclassified as severe drops.

10.4 IST-Safe Date Resolution

All scheduled queries explicitly compute the processing date using:

```
DATE_SUB(CURRENT_DATE('Asia/Kolkata'), INTERVAL 1 DAY)
```

Why Not Use CURRENT_DATE()

- `CURRENT_DATE()` defaults to UTC
- UTC midnight does not align with IST business days
- Causes date drift during early morning hours

Using `CURRENT_DATE('Asia/Kolkata')` guarantees that calendar boundaries match business expectations.

10.5 Scheduled Query Ordering

The pipeline relies on a strict execution order to preserve data dependencies. Each job is scheduled with sufficient buffer time to ensure upstream completion.

10.5.1 Daily Schedule (UTC)

Time (UTC)	Job	Purpose
19:00	Synthetic GA4 Generator	Create clean daily events
19:10	Anomaly Injector	Inject probabilistic anomalies
20:30	Metric Aggregation	Aggregate daily metrics
20:35	Gap Filling	Ensure metric continuity
20:45	Anomaly Detection	Forecast and classify anomalies
20:50	Severity and Business Logic	To assign categories
20:55	Alert Decision	For selecting only high/critical categories
21:00	Email Script	For sending final output

Why Staggered Execution

- Avoids race conditions
- Ensures downstream queries see complete upstream outputs
- Allows easy debugging by isolating failures

10.6 Idempotent Daily Processing

Each scheduled query follows a strict idempotency pattern:

1. Compute `run_date`
2. Delete existing rows for that date
3. Insert freshly computed results

Benefits

- Safe re-runs
- No duplicate records
- Deterministic outputs

This design enables both automatic retries and manual backfills without code changes.

10.7 Replay and Backfill Support

The pipeline includes a control mechanism via the `pipeline_runtime` table, allowing operators to override the default date logic.

Replay Mode When `replay_mode = TRUE`, the pipeline processes a user-specified date instead of "yesterday".

Use Cases

- Historical backfills
- Incident recovery
- Model retraining alignment

This mechanism eliminates the need to modify SQL code for non-standard runs.

10.8 Failure Isolation and Recovery

Because each day is processed independently:

- A failure on one day does not affect other days
- Partial pipeline failures can be re-run safely
- Downstream tables remain consistent

In the event of a failure:

1. The scheduled query logs capture the error
2. The failed date can be replayed
3. All dependent outputs are recomputed deterministically

10.9 Why This Scheduling Model Is Production-Grade

This scheduling and timezone strategy mirrors best practices used in enterprise analytics systems:

- Clear separation of execution time and business logic
- Timezone-safe date handling
- Deterministic, idempotent batch processing
- Built-in replay and recovery mechanisms

As a result, the pipeline remains robust, auditable, and scalable as data volume and model complexity increase.

11 Severity Classification, Business Impact, Root Cause, and Recommendations

This section documents the complete decision logic used to translate statistical anomalies into business-interpretable outcomes. The logic implemented here is **fully deterministic, metric-specific, and strictly aligned with the statistical anomaly detection layer**.

The framework recognizes that not all metric deviations carry equivalent business risk. A revenue decline represents an immediate threat requiring escalation, while a revenue spike typically indicates positive performance or data quality issues rather than operational crisis. Conversely, traffic and engagement metrics behave bidirectionally—both unexpected surges and collapses warrant investigation as they may indicate bot activity, tracking failures, or viral events.

All rules described below are implemented verbatim in the production SQL query and operate only on materialized outputs from the statistical anomaly table.

—

11.1 Input Signals

Each event-day record entering this stage contains the following statistical signals:

- **actual_value**: Observed metric value
- **expected_value**: Forecasted value from ARIMA_PLUS
- **lower_bound, upper_bound**: Prediction interval bounds
- **deviation_pct**: Relative deviation from forecast
- **z_score**: Standardized deviation
- **anomaly_probability**: ML anomaly confidence
- **is_outside_bounds**: Interval breach indicator
- **is_anomaly**: ML anomaly flag

No business logic is applied if `is_anomaly = FALSE` or `is_outside_bounds = FALSE`.

—

11.2 Step 1: Severity Classification

Severity represents **statistical urgency**—the magnitude of deviation from expected behavior that demands immediate attention. Classification follows a strict, metric-aware hierarchy that distinguishes between **decline-only monitoring** for revenue-critical events and **bidirectional monitoring** for traffic and behavioral signals.

11.2.1 Hard Gating

All severity evaluation begins with a mandatory statistical gate:

- If `is_anomaly = FALSE` OR `is_outside_bounds = FALSE`
- Then **severity_level = NONE**

This ensures that no business interpretation occurs without underlying statistical justification. The ML model and prediction intervals must agree that an anomaly exists before any severity assignment proceeds.

—

11.2.2 Directional Logic: Decline-Only vs. Bidirectional Monitoring

Decline-Only Events (Purchase, Add to Cart, Add Payment Info) For metrics directly tied to revenue and conversion completion, severity classification activates exclusively on **negative deviations**—situations where observed values fall significantly below forecasted expectations. This reflects the asymmetric business reality that while revenue collapses require immediate intervention, revenue spikes (whether from legitimate bulk orders, data duplication, or tracking errors) do not represent operational crises demanding emergency response.

The severity determination for decline-only events evaluates both the **magnitude of underperformance** (how far below forecast) and the **statistical extremity** (how unusual this deviation is relative to historical variance). For instance, a purchase event showing deviation substantially below forecast combined with a highly negative z-score would escalate to CRITICAL severity, indicating potential payment system failure or severe checkout friction. The same magnitude in the positive direction would not trigger severity classification, as it represents business upside or, at worst, data quality issues resolvable during normal business hours.

Bidirectional Events (Page View, Session Start, User Engagement) Traffic acquisition and behavioral engagement metrics operate under **bidirectional monitoring**—both significant collapses and unexpected surges warrant severity classification. This reflects the operational reality that traffic anomalies in either direction indicate potential system issues: collapses suggest acquisition failures or tracking breakage, while spikes often signal bot traffic, security incidents, or data duplication that can distort analytics and strain infrastructure.

For bidirectional events, the severity logic evaluates absolute deviation from forecast regardless of direction. A severe traffic drop might indicate paused campaigns or SEO failures, while an equivalent surge might reveal bot infiltration or duplicate event firing in single-page applications. Both scenarios require investigation, though the suspected root causes and recommended actions differ based on the directionality of the deviation.

—

11.2.3 Severity Hierarchy

Within each event type, severity follows a graduated escalation ladder:

- **CRITICAL**: Indicates fundamental system failure or severe business impact requiring immediate emergency response. For decline-only events, this captures catastrophic revenue loss scenarios such as complete payment gateway outages or checkout system failures. For bidirectional events, this encompasses both total traffic acquisition collapse and extreme artificial inflation indicating security or data integrity crises.

- **HIGH**: Represents significant deviation from expected behavior demanding urgent investigation but not immediate emergency escalation. For revenue events, this captures substantial conversion degradation. For traffic events, this includes meaningful drops or spikes that may indicate campaign issues, bot activity, or tracking degradation.
- **MEDIUM**: Moderate statistical anomalies with potential business relevance. These situations warrant monitoring and investigation during regular business operations but do not require after-hours escalation.
- **LOW**: Statistical anomalies of limited magnitude or confidence. These are logged for pattern analysis but do not trigger active response.
- **NONE**: No statistical anomaly detected; business logic does not execute.

The transition between severity levels is determined by combined thresholds of deviation magnitude and statistical confidence. For example, a purchase event might escalate from HIGH to CRITICAL when the negative deviation exceeds a substantial percentage of forecasted value **and** the z-score indicates extreme statistical rarity—suggesting not merely a bad day but a system-level failure.

—

11.3 Business Impact Classification

Business impact represents **organizational consequence**—the translation of statistical severity into prioritization frameworks that guide resource allocation and response urgency. While severity captures statistical extremity, impact captures the actual or potential damage to business operations.

The impact classification maintains the same directional logic as severity: revenue-critical events evaluate negative deviations only, while traffic and engagement events evaluate absolute deviations.

Revenue Impact (Purchase Events) Purchase anomalies carry the highest potential business consequence. The impact classification distinguishes between severe revenue collapse indicating immediate financial threat and general revenue underperformance requiring attention. A severe negative deviation—where actual revenue collapses to a small fraction of forecasted values with high statistical confidence—triggers VERY_HIGH impact, indicating potential platform failure where customers cannot complete transactions. Less severe but still significant negative deviations trigger HIGH impact, suggesting conversion friction or demand degradation that impacts financial performance without indicating complete system failure.

Funnel Impact (Add to Cart, Add Payment Info) Upper-funnel conversion events carry HIGH impact when they show significant negative deviations,

as these represent early indicators of downstream revenue collapse. A substantial drop in add-to-cart rates signals that product page traffic is failing to convert into purchase intent, while add-payment-info declines indicate checkout friction. These are classified as HIGH impact because they predict revenue degradation and require immediate funnel optimization to prevent downstream purchase collapse.

Traffic and Engagement Impact Page views, session starts, and user engagement carry MEDIUM impact when showing significant absolute deviations. While traffic collapses or spikes indicate operational issues requiring investigation, they do not directly translate to immediate revenue loss in the way purchase failures do. The MEDIUM classification reflects that these situations demand attention and diagnostic effort but may represent campaign adjustments, seasonal patterns, or tracking issues rather than business-critical failures.

—

11.4 Root Cause Identification

Root cause analysis translates statistical anomalies into human-interpretable explanations of underlying business or technical drivers. This layer executes against the severity and impact classifications, providing context that guides investigation and remediation.

The root cause logic operates through **deviation-band stratification**—partitioning the range of possible deviation magnitudes into distinct scenarios with characteristic signatures and likely explanations. Rather than treating all negative deviations as equivalent, the logic recognizes that a slight revenue underperformance suggests different root causes than a catastrophic collapse.

Stratification Logic For decline-only events, the stratification progresses from severe collapse (indicating system failure) through moderate degradation (suggesting friction or demand issues) to early-stage weakening (pointing to optimization opportunities). For example, in purchase events, extreme negative deviations typically indicate payment infrastructure failure or complete checkout breakage, while moderate deviations suggest campaign misalignment or competitive pressure, and smaller deviations point to UX friction or performance regression.

For bidirectional events, the stratification applies separately to negative and positive deviations. Negative deviations progress from total acquisition failure (tracking collapse or campaign shutdown) through partial loss (channel degradation or consent blocking) to moderate decline (bid adjustments or content changes). Positive deviations progress from artificial inflation (bots or duplicate firing) through coordinated surges (campaigns or viral events) to moderate growth (seasonal uplift or optimization success).

Each deviation band carries a distinct narrative explanation grounded in operational experience. These explanations describe not merely what statistically

occurred but what business scenarios typically produce such patterns—enabling investigators to rapidly narrow hypothesis spaces and focus diagnostic efforts on high-probability causes.

If no stratification condition matches, the system assigns a default indicating that the deviation pattern does not match known signatures, requiring custom investigation.

—

11.5 Recommended Actions

For every identified root cause scenario, the system generates exactly two recommended actions: one focused on technical validation or data investigation, and one focused on user experience, behavioral analysis, or business process review. This pairing ensures that investigations cover both the mechanical correctness of systems and the human experience of users.

The recommendations are calibrated to the specific scenario—extreme collapse triggers infrastructure verification and emergency testing, while moderate degradation triggers performance review and competitive analysis. For artificial inflation scenarios, recommendations focus on bot detection and deduplication validation; for legitimate growth, recommendations focus on attribution confirmation and trend monitoring.

If no root cause scenario matches, the system returns default null indicators, prompting custom investigation beyond standard playbooks.

—

11.6 Step 1: Severity Classification

11.6.1 Critical Severity Thresholds

CRITICAL — Purchase (Negative Declines Only)

- Deviation $\leq -60\%$ OR z-score ≤ -3.5

CRITICAL — Add to Cart (Negative Declines Only)

- Deviation $\leq -80\%$ OR z-score ≤ -8.0

CRITICAL — Add Payment Info (Negative Declines Only)

- Deviation $\leq -60\%$ OR z-score ≤ -5.0

CRITICAL — Page View (Bidirectional)

- Negative: Deviation $\leq -50\%$ OR z-score ≤ -5.5
- Positive: Deviation $\geq +60\%$ OR z-score $\geq +5.5$

CRITICAL — Session Start (Bidirectional)

- Negative: Deviation $\leq -45\%$ OR z-score ≤ -6.5
- Positive: Deviation $\geq +55\%$ OR z-score $\geq +6.5$

CRITICAL — User Engagement (Bidirectional)

- Negative: Deviation $\leq -60\%$ OR z-score ≤ -4.0
- Positive: Deviation $\geq +70\%$ OR z-score $\geq +4.0$

—

11.6.2 High Severity Thresholds

HIGH — Purchase (Negative Declines Only)

- Deviation $\leq -30\%$ OR z-score ≤ -2.0

HIGH — Add to Cart (Negative Declines Only)

- Deviation $\leq -40\%$ OR z-score ≤ -4.0

HIGH — Add Payment Info (Negative Declines Only)

- Deviation $\leq -40\%$ OR z-score ≤ -3.5

HIGH — Page View (Bidirectional)

- Negative: Deviation $\leq -35\%$ OR z-score ≤ -4.0
- Positive: Deviation $\geq +45\%$ OR z-score $\geq +4.0$

HIGH — Session Start (Bidirectional)

- Negative: Deviation $\leq -25\%$ OR z-score ≤ -4.0
- Positive: Deviation $\geq +35\%$ OR z-score $\geq +4.0$

HIGH — User Engagement (Bidirectional)

- Negative: Deviation $\leq -35\%$ OR z-score ≤ -3.5
- Positive: Deviation $\geq +45\%$ OR z-score $\geq +3.5$

—

11.6.3 Medium and Low Severity

- **MEDIUM**: Absolute deviation between 10%-30% OR (anomaly probability ≥ 0.85 OR $|z| \geq 1.5$)
- **LOW**: Residual anomalies not meeting escalation criteria

—

11.7 Step 2: Business Impact Classification

Business impact represents **organizational consequence**, with directional logic aligned to severity rules.

11.7.1 Purchase Impact (Negative Only)

- **VERY_HIGH**: Deviation $\leq -60\%$ OR z-score ≤ -3.5
- **HIGH**: Any other purchase anomaly (negative deviation)

11.7.2 Funnel Impact (Negative Only)

- **HIGH**: Deviation $\leq -30\%$ OR z-score ≤ -3.0 (for add_to_cart, add_payment_info)

11.7.3 Traffic and Engagement Impact (Bidirectional)

- **MEDIUM**: Absolute deviation $\geq 30\%$ OR $|z| \geq 3.0$ (for page_view, session_start, user_engagement)
- **LOW**: Otherwise

—

11.8 Step 3: Root Cause Identification

Root cause analysis executes against **deviation-specific bands** to provide granular, actionable explanations. Logic branches on both event type and magnitude/direction of deviation.

11.8.1 Purchase Root Causes (Negative Only)

- **Deviation** $\leq -90\%$: Regional payment gateway failures, currency conversion bugs, fraud-prevention overblocking, geo-specific outages
- **Deviation** -90% **to** -60%: Checkout degradation—shipping charge surprises, tax errors, broken discounts, forced account creation, tightened fraud thresholds
- **Deviation** -60% **to** -45%: Campaign misalignment, broken landing pages, pricing promise mismatches, targeting drift

- **Deviation** -45% **to** -30%: Early-stage degradation—page load regressions, mobile UX issues, trust signal removal, delivery promise changes

11.8.2 Add to Cart Root Causes (Negative Only)

- **Deviation** $\leq -80\%$: Behavioral deterrents—price shocks, discount errors, shipping fee visibility, broken images, stock indicator bugs
- **Deviation** -80% **to** -60%: Traffic quality degradation, broad targeting, curiosity clicks without purchase intent
- **Deviation** -60% **to** -40%: Early funnel softening—slower load times, mobile performance, reduced promotional urgency, competitor pricing

11.8.3 Add Payment Info Root Causes (Negative Only)

- **Deviation** $\leq -90\%$: Partial payment step failure (mobile/checkout-specific), trust erosion from fee additions or security warnings
- **Deviation** -90% **to** -60%: Checkout usability degradation— long forms, auto-focus bugs, poor error handling, accessibility regressions
- **Deviation** -60% **to** -40%: Behavioral hesitation—price sensitivity, low urgency, external comparison behavior

11.8.4 Page View Root Causes (Bidirectional)

Negative Deviations:

- $\leq -75\%$: Total analytics instrumentation failure, campaign pauses, revoked ad accounts, budget exhaustion, 404/500 errors
- -75% **to** -50%: Partial traffic loss—SEO collapses, region-specific consent blocking, CDN/firewall rules
- -50% **to** -35%: Marketing changes—paused remarketing, reduced bids, content removals, broken internal links

Positive Deviations:

- $\geq +90\%$: Bot traffic, scrapers, DDoS floods, misconfigured uptime monitors, duplicate page_view firing
- $+90\%$ **to** $+60\%$: Flash sales, influencer campaigns, viral content (if supported by marketing), or crawler/referral spam
- $+60\%$ **to** $+45\%$: Healthy growth, seasonal uplift, promotions, or early-stage bot activity

11.8.5 Session Start Root Causes (Bidirectional)

Negative Deviations:

- $\leq -75\%$: CMP misconfiguration, consent blocking without Consent Mode v2, redirect chains stripping session parameters
- -75% **to** -45%: Partial suppression—device-specific issues, Safari ITP, conditional GTM trigger failures
- -45% **to** -25%: Attribution changes, UTM structure modifications, server-side tagging issues, shortened timeout windows

Positive Deviations:

- $\geq +80\%$: Session fragmentation—cookie persistence failures, SameSite misconfigurations, cross-domain navigation issues
- $+80\%$ **to** $+55\%$: SPA routing issues, misfiring session_start on route changes
- $+55\%$ **to** $+35\%$: Legitimate re-engagement or early fragmentation signs

11.8.6 User Engagement Root Causes (Bidirectional)

Negative Deviations:

- $\leq -85\%$: Critical performance failure—LCP > 4s, CLS > 0.25, broken engagement instrumentation
- -85% **to** -60%: Partial suppression—mobile layout failures, cookie banner/modal overlays blocking interaction
- -60% **to** -35%: Content relevance issues—template redesigns pushing content below fold, diluted information density

Positive Deviations:

- $\geq +90\%$: Artificial inflation—bot JavaScript execution, QA automation, SPA timer misconfiguration
- $+90\%$ **to** $+70\%$: Misfiring timers—Page Visibility API misuse, background tab accumulation
- $+70\%$ **to** $+45\%$: Legitimate content success or early inflation indicators

If no condition matches: `No dominant root cause identified`.

—

11.9 Step 4: Recommended Actions

For every identified root cause band, the system assigns **exactly two** recommended actions:

- One technical or segmentation analysis step
- One behavioral, UX, or validation investigation step

Actions are calibrated to the specific deviation magnitude and direction. If no root cause is identified, default actions return `['NA', 'NA']`.

11.10 Guarantees Provided by This Layer

This stage guarantees that:

- No alert is escalated without statistical justification
- Directional logic respects business context—negative-only for revenue events, bidirectional for traffic and engagement
- Root causes are deviation-band-specific and never contradict severity logic
- Recommendations are deterministic, auditable, and calibrated to magnitude
- Business interpretation remains explainable and reproducible

12 Alert Eligibility and Suppression Logic

12.1 Why an Alert Decision Layer Is Required

Severity classification identifies how abnormal and how impactful an anomaly is, but **severity alone does not imply that an alert should be sent**.

In production monitoring systems, uncontrolled alerting leads to:

- Alert fatigue among stakeholders
- Loss of trust in monitoring systems
- Ignored or delayed response to real incidents

Therefore, the pipeline introduces a dedicated **alert eligibility and suppression layer** that determines whether a scored anomaly warrants human notification.

This layer represents the final decision boundary between automated analytics and human intervention.

—

12.2 Separation of Concerns

This stage operates strictly **after** severity and business impact classification and follows two rules:

- It does not re-evaluate statistical anomaly logic
- It does not modify severity or impact classifications

Its sole responsibility is to decide:

Should a human be notified about this anomaly right now?

—

12.3 Alert Eligibility Criteria

An anomaly is considered **eligible for alerting** if and only if it satisfies one of the following conditions:

$$is\_anomaly = TRUE \land severity = CRITICAL$$

$$\lor (severity = HIGH \land business\_impact \in \{HIGH, VERY\_HIGH\})$$

Rationale

- **Critical severity override**: CRITICAL anomalies always represent immediate risk requiring human attention
- **High severity gating**: HIGH severity alerts are restricted to high business impact scenarios only
- **Business alignment**: Ensures HIGH severity alerts represent genuine operational risk, not just statistical deviations

This combination prevents alerts for:

- Noisy traffic metrics
- Low-impact engagement anomalies
- Statistically interesting but operationally irrelevant events
- HIGH severity anomalies with low business impact (LOW/MEDIUM)

—

12.4 Repeated Alert Suppression

Repeated alerts for the same issue can rapidly degrade alert effectiveness.

To address this, the system checks whether an alert of the same `event_name` and `severity_level` was raised on the previous day.

Repeated Alert Definition An alert is considered repeated if:

- The same metric triggered an eligible alert yesterday
- The severity level is unchanged

Rationale

- Persistent issues should be tracked, not repeatedly alerted
- Reduces alert spam for long-running incidents
- Encourages resolution rather than notification loops

Note on CRITICAL Alerts While repeated CRITICAL alerts are identified for audit purposes, they are never suppressed. CRITICAL severity indicates immediate business risk that requires continuous human awareness until resolved.

—

12.5 Suppression Rules

Once eligibility is established, suppression rules are applied in priority order:

1. If the anomaly is not alert-eligible → suppress
2. If severity is **CRITICAL** → never suppress (override)
3. If alert is repeated → suppress (non-CRITICAL only)

Critical Override CRITICAL anomalies represent immediate business risk and override all suppression logic, including repetition checks. This ensures that critical incidents are never missed due to deduplication logic.

—

12.6 Alert Priority Assignment

Each non-suppressed alert is assigned a priority to guide response urgency:

- **P0**: CRITICAL severity (immediate action required)
- **P1**: HIGH severity with HIGH or VERY_HIGH business impact
- **NONE**: Informational, suppressed, or ineligible alerts

This mapping enables consistent escalation and integration with incident management systems. P0 alerts bypass standard triage and require immediate on-call response.

12.7 Behavioral Guarantees

This alert decision framework guarantees that:

- **CRITICAL anomalies are always alerted** (P0) regardless of repetition or other factors
- **HIGH anomalies are conditional** on having HIGH or VERY_HIGH business impact (P1)
- **Low-impact anomalies are suppressed** to prevent alert fatigue
- **Repeated non-critical alerts are suppressed** to reduce noise from persistent issues
- **Decisions are deterministic and auditable** through the explicit eligibility and suppression logic

This table serves as the authoritative source for all downstream alert delivery mechanisms, ensuring that stakeholders receive only actionable, high-signal notifications.

—

12.8 Operational Considerations

No Business Hours Restrictions Unlike previous iterations, this version removes business hours awareness. CRITICAL alerts may arrive at any time, reflecting their genuine urgency. This design assumes:

- 24/7 on-call coverage for P0 incidents
- P1 alerts (HIGH + high impact) are important but do not require the same immediate response as CRITICAL

Deduplication Scope The repeated alert check uses a 24-hour window and exact severity matching. This means:

- If severity escalates from HIGH to CRITICAL, a new alert fires
- If severity de-escalates, the change is captured as a new event
- Persistent issues must be resolved to reset the deduplication window

13 Email Payload Construction

13.1 Why an Email Payload Layer Is Required

After alert eligibility and suppression decisions are finalized, the system must deliver alerts to stakeholders in a structured, human-readable format. Directly

embedding formatting logic inside the delivery mechanism (Apps Script) would tightly couple alert content with delivery code, making the system harder to maintain and audit.

To avoid this, the pipeline introduces a dedicated **email payload construction layer** implemented as a BigQuery view. This layer acts as a semantic contract between analytics logic and alert delivery.

Key Principle

> *All business logic lives in BigQuery. The delivery layer only renders what it receives.*

—

13.2 Why a View Instead of a Table

The email payload is implemented as a **view**, not a materialized table, for the following reasons:

- Alert eligibility may change if upstream logic is re-run
- Views always reflect the latest alert decisions
- No risk of stale or duplicated alert records
- No additional storage or scheduling overhead

This design ensures that email delivery always operates on the most current and authoritative alert state.

—

13.3 Input Tables

The email payload view consumes two upstream tables:

- `ga4_anomaly_scored_events` — provides metric context and severity
- `ga4_anomaly_alert_decisions` — provides alert eligibility and suppression

Joining these tables ensures that:

- Only approved alerts are delivered
- Full diagnostic context is preserved

—

13.4 Filtering Strategy

The view applies a strict filter:

$$alert\_eligible = TRUE \wedge suppressed = FALSE$$

Rationale This guarantees that:

- No suppressed alerts are accidentally sent
- Delivery logic never re-evaluates alert conditions
- Exactly-once alert semantics are preserved

—

13.5 Field Design and Rationale

Each column in the email payload serves a specific purpose.

13.5.1 Email Routing Fields

- `email_from`, `email_to`, `email_cc`

These fields are defined in BigQuery to allow future extensions such as:

- Per-client routing
- Severity-based escalation lists
- Team-specific notifications

—

13.5.2 Identity and Context Fields

- Client identifier
- GA4 property identifier
- Metric name
- Industry
- Alert date and timezone

These fields ensure that alert recipients can immediately contextualize the incident without external lookup.

—

13.5.3 Observed vs Expected Metrics

The view includes:

- Actual observed value
- Forecasted expected value
- Prediction interval bounds

Deviation Formatting Deviation percentage is pre-formatted in the view:

- If expected value is zero, deviation is rendered as `N/A (no reliable baseline)`
- Otherwise, deviation is converted to a percentage string

This avoids conditional formatting logic in the delivery layer.

—

13.5.4 Severity and Impact Fields

Severity level, business impact, and anomaly score are included verbatim from upstream tables. This ensures that:

- Alert urgency is explicit
- Escalation decisions are transparent
- Emails remain auditable

—

13.5.5 Root Cause and Supporting Signals

In the current implementation, root cause fields are static placeholders. However, they are explicitly modeled to support future enhancements such as:

- Automated root cause inference
- Metric correlation analysis
- External system annotations

—

13.5.6 Benchmark and Evaluation Fields

Industry benchmark thresholds and evaluation flags are embedded to:

- Provide external context to anomaly magnitude
- Justify escalation decisions
- Improve stakeholder trust in alerts

—

13.5.7 Recommended Actions

Recommended actions are represented as an array to allow:

- Multiple actionable steps
- Dynamic rendering in email templates
- Future role-based action suggestions

—

13.6 Email Payload View SQL

13.6.1 Production Email Payload View

```
-- EXACT PRODUCTION LOGIC (VERBATIM)
CREATE OR REPLACE VIEW
‘tvc-ecommerce.analytics_live.ga4_anomaly_email_payload_view‘
AS
SELECT
  -- Email routing
  ’dhananjay@tatvic.com’ AS email_from,
  ’ronit@tatvic.com’ AS email_to,
  [’aarya@tatvic.com’,’vishnu@tatvic.com’] AS email_cc,

  -- Identity
  ’tvc-ecommerce’ AS client_id,
  ’GA4-Ecommerce-Prod’ AS ga4_property_id,
  s.event_name AS metric_name,
  ’Ecommerce’ AS industry,
  s.event_date AS alert_date,
  ’Asia/Kolkata’ AS timezone,

  -- Observed vs Expected
  s.actual_value,
  s.expected_value,
```

```
  s.lower_bound,
  s.upper_bound,
  CASE
    WHEN s.expected_value = 0 THEN 'N/A (no reliable baseline)'
    ELSE CONCAT(ROUND(s.deviation_pct * 100, 2), '%')
  END AS deviation_display,

  -- Severity
  s.severity_level,
  s.business_impact,
  s.anomaly_probability AS anomaly_score,

  -- Root cause (static v1)
  'CHECKOUT_OR_PAYMENT_FAILURE' AS suspected_cause_category,
  'HIGH' AS root_cause_confidence,
  [
    'Concurrent conversion anomaly detected',
    'Traffic remains stable',
    'Revenue deviation exceeds industry threshold'
  ] AS supporting_signals,

  -- Benchmarks
  25 AS medium_deviation_pct,
  50 AS critical_deviation_pct,
  'CRITICAL_BREACH' AS benchmark_status,

  -- Alert evaluation
  a.within_business_hours,
  FALSE AS known_event_detected,
  a.suppressed,

  -- Recommended actions
  [
    'Verify payment gateway status',
    'Check GA4 purchase event firing',
    'Validate transaction_id duplication'
  ] AS recommended_actions

FROM `tvc-ecommerce.analytics_live.ga4_anomaly_alert_decisions` a
JOIN `tvc-ecommerce.analytics_live.ga4_anomaly_scored_events` s
USING (event_date, event_name)
WHERE
  a.alert_eligible = TRUE
  AND a.suppressed = FALSE;
```

—

13.7 Guarantees Provided by This Layer

This email payload layer guarantees that:

- Only finalized alerts are delivered
- Delivery code remains logic-free
- Alert content is deterministic and reproducible
- The system is extensible without refactoring Apps Script

This view represents the final analytical contract consumed by the alert delivery mechanism.

14 Automated Alert Delivery via Google Apps Script

14.1 Purpose of the Delivery Layer

The objective of the delivery layer is to notify stakeholders about business-critical anomalies identified by the analytics pipeline. This layer represents the final step in the anomaly detection workflow and serves as the interface between automated analytics and human response.

Crucially, the delivery layer is designed to be **logic-free**. All decisions regarding anomaly detection, severity classification, business impact, eligibility, and suppression are finalized upstream inside BigQuery.

The Apps Script layer therefore performs only three responsibilities:

1. Refresh the email payload view
2. Fetch finalized alert records
3. Send notification emails

—

14.2 Why Google Apps Script

Google Apps Script is selected for alert delivery due to:

- Native integration with Gmail
- First-class support for BigQuery APIs
- Built-in authentication and authorization
- Time-driven execution without infrastructure management

This enables a fully serverless, low-maintenance alerting solution without external services or credentials.

—

14.3 Why Apps Script Contains No Business Logic

All business logic is intentionally implemented in BigQuery for the following reasons:

- BigQuery logic is declarative and auditable
- SQL-based decisions are reproducible and replayable
- Business rules can be validated independently of delivery
- Separation of concerns simplifies operational debugging

As a result, Apps Script does not:

- Evaluate anomaly thresholds
- Determine severity or business impact
- Decide whether an alert should be suppressed

It only consumes the final, authoritative output.

—

14.4 Why Only Certain Anomalies Trigger Emails

Emails are sent only for anomalies that satisfy all of the following conditions upstream:

$$severity \in \{HIGH, CRITICAL\} \land business\_impact \in \{HIGH, VERY\_HIGH\} \land alert\_eligible = TRUE \land sup$$

Rationale

- Prevents alert fatigue
- Ensures notifications represent material business risk
- Avoids escalation of diagnostic or low-impact anomalies

This design guarantees that emails correspond exclusively to actionable incidents.

—

14.5 Execution Model

The Apps Script is executed using a time-driven trigger scheduled after all BigQuery scheduled queries have completed.

- Trigger type: Daily
- Execution time: After alert decision materialization
- Trigger guarantees exactly-once execution per day

No deployment step is required because the script is not exposed as a web application or external API.

—

14.6 Email Generation Strategy

Each qualifying alert row generates exactly one email.

Why One Email per Metric

- Preserves clarity and focus
- Avoids conflating unrelated anomalies
- Enables independent resolution tracking

If multiple metrics trigger alerts on the same day, multiple emails are sent, each corresponding to a distinct business issue.

—

14.7 Email Subject Design

The email subject line embeds severity and metric name:

```
[SEVERITY] GA4 Anomaly Detected { <metric_name>
```

This allows recipients to:

- Identify urgency without opening the email
- Apply inbox rules or escalation workflows

14.8 Delivery Guarantees

This delivery mechanism guarantees:

- Exactly-once email delivery per metric per day
- Zero business logic duplication
- Deterministic, auditable alerting
- Fully automated, serverless execution

This completes the end-to-end anomaly detection and alerting pipeline.