Z-SCORE MODEL SPECIFICATION

1. Purpose of the model

This model is designed to detect anomalies where a player's **performance rating** significantly exceeds their **initial rating**, by comparing each player's rating difference against the distribution of differences **within the same tournament dataset** (same event/category/session group).

2. Inputs (per player *i*)

For each player *i*:

- **Rating** – player's rating
- **PerformanceRating** – player's performance rating
- **Difference** – rating difference
- **Difference = PerformanceRating − Rating**

All calculations must be performed **only within the same grouping** (same event/category/location/session as agreed).

---

3. Global statistical parameters (computed for the whole group)

Assume there are **N** players in the group.

3.1 Mean (μΔ)

$$\mu_\Delta = \frac{1}{N}\sum Difference_i$$

Excel (if Difference is AC2:AC999):
```
=AVERAGE($AC$2:$AC$999)
```

SQL:
```
AVG(CAST(Difference AS float)) OVER (PARTITION BY EventId, CategoryId)
```

3.2 Standard deviation (σΔ)

Use **population** standard deviation (not sample):

$$\sigma_\Delta = \sqrt{\frac{1}{N}\sum(Difference_i - \mu_\Delta)^2}$$

Excel:
```
=STDEV.P($AC$2:$AC$999)
```

```
SQL SQRT(
  AVG(CAST(Difference AS float) * CAST(Difference AS float)) OVER (PARTITION
BY EventId, CategoryId)
  - POWER(AVG(CAST(Difference AS float)) OVER (PARTITION BY EventId,
CategoryId), 2)
)
```

---

4. Z-Score calculation (core formula)

For each player:

$$Z = \frac{Difference - \mu_\Delta}{\sigma_\Delta}$$

Excel (AC2=Difference, μΔ in AG1, σΔ in AH1):
```
=(AC2-$AG$1)/$AH$1
```
or:
```
=(AC2-AVERAGE($AC$2:$AC$999))/STDEV.P($AC$2:$AC$999)
```

SQL:

```
CASE
  WHEN sigma = 0 OR sigma IS NULL THEN NULL
  ELSE (Difference - mu) / sigma
END
```

---

5. Probability (p-value) calculation

A key logical nuance applies here.

Since the model is designed to detect **unusually large positive deviations** (i.e., very high Difference values), a **one-sided right-tail probability** must be used.

The correct formula is:

p=1−Φ(Z)p = 1 - \Phi(Z)p=1−Φ(Z)

where:

- Φ(Z)\Phi(Z)Φ(Z) is the cumulative distribution function (CDF) of the standard normal distribution
- Z is the computed Z-score

This evaluates the probability of observing a value at least as large as Z under the normal distribution.

Excel (one-sided, right tail):
```
=1-NORM.S.DIST(Z2,TRUE)
```

SQL (variant A: if you have NormalCdf UDF):
```
p = 1.0 - dbo.NormalCdf(ZScore)
```

SQL (variant B: using ERF, if supported):
p=1.0 - (0.5 * (1.0 + ERF(ZScore / SQRT(2.0))))

---

6. Anomaly classification rules

Use **Zscore**::

- **Z < 2.0** → NORMAL
- **2.0 ≤ Z < 3.0** → HIGH
- **3.0 ≤ Z < 3.5** → VERY HIGH
- **Z ≥ 3.5** → ANOMALY

Excel (assuming Z in AG2):
```
=IF(AG2>=3.5,"ANOMALY",IF(AG2>=3.0,"VERY HIGH",IF(AG2>=2.0,"HIGH","NORMAL")))
```

SQL:

```
CASE
  WHEN Z >= 3.5 THEN 'ANOMALY'
  WHEN Z >= 3.0 THEN 'VERY HIGH'
  WHEN Z >= 2.0 THEN 'HIGH'
  ELSE 'NORMAL'
END
```

---

7. Calculation scope (mandatory)

μΔ and σΔ must be computed **only within the same group** as the player:

- same contest/event
- same category
- same location (if used)

Do **not** mix different events or seasons.

---

8. Typical implementation steps

1. Compute **Difference**
2. Compute group **μΔ**
3. Compute group **σΔ**
4. Compute **ZScore**
5. Compute **Probability (one-sided)**
6. Assign **AnomalyClass**

---

9. Stability and data rules

-Zero Standard Deviation Case

If: $\sigma\Delta = 0$ then:

- ZScore shall be set to **NULL**
- AnomalyClass shall be set to **NORMAL**

Explanation:
Zero standard deviation means that all Difference values within the group are identical.
Therefore, no statistical outlier can exist.

-Half or Non-Active Rating Flags (h / n/a)

If a rating is marked as:

- half (h)
- non-active (n/a)

the numeric rating value shall be used without modification.

The flags do not affect the calculation.

Example:
2150h → treated as 2150
2100 n/a → treated as 2100

-Missing Initial Rating (First-Time Participant)

If a player has no initial rating:

Rating shall be set equal to PerformanceRating.

This ensures:

Difference = PerformanceRating − Rating = 0

This prevents artificial inflation of ZScore for first-time participants.

Technically Correct Final Version

• If $\sigma\Delta = 0$ → ZScore = NULL, classify as NORMAL.
• Rating flags (h / n/a) do not modify the numeric rating value.
• If Rating is NULL → set Rating = PerformanceRating (Difference = 0).

SQL VIEW (with one-sided p-value and rating rules)

```
CREATE OR ALTER VIEW dbo.vw_Isc_ZScoreModel
AS
WITH base AS (
    SELECT
        r.PlayerId,
        r.PlayerName,

        -- GROUP KEYS (MUST MATCH YOUR BUSINESS LOGIC)
        r.ContestId,
        r.LocationId,
        r.CategoryId,

        -- Rating rules:
        -- If Rating is missing (first-time), treat it as PerformanceRating
(Difference=0)
        CAST(COALESCE(r.Rating, r.PerformanceRating) AS float) AS Rating,
        CAST(r.PerformanceRating AS float) AS PerformanceRating,

        CAST(r.PerformanceRating - COALESCE(r.Rating, r.PerformanceRating) AS
float) AS Difference
    FROM dbo.IscResults r
    WHERE r.PerformanceRating IS NOT NULL
),
stats AS (
    SELECT
        b.*,

        AVG(b.Difference) OVER (
            PARTITION BY b.ContestId, b.LocationId, b.CategoryId
        ) AS MuDiff,

        -- Population SD: sqrt(E[x^2] - (E[x])^2)  (matches Excel STDEV.P)
        SQRT(
            AVG(b.Difference * b.Difference) OVER (
                PARTITION BY b.ContestId, b.LocationId, b.CategoryId
            )
            - POWER(
                AVG(b.Difference) OVER (
                    PARTITION BY b.ContestId, b.LocationId, b.CategoryId
                ),
                2
            )
        ) AS SigmaDiff
    FROM base b
),
zcalc AS (
    SELECT
        s.*,

        CASE
            WHEN s.SigmaDiff IS NULL OR s.SigmaDiff = 0 THEN NULL
            ELSE (s.Difference - s.MuDiff) / s.SigmaDiff
        END AS ZScore
    FROM stats s
)
SELECT
    z.PlayerId,
    z.PlayerName,

    z.ContestId,
    z.LocationId,
```

```
    z.CategoryId,

    z.Rating,
    z.PerformanceRating,

    z.Difference,
    z.MuDiff,
    z.SigmaDiff,
    z.ZScore,

    -- One-sided right-tail probability: p = 1 - Phi(Z)
    CASE
        WHEN z.ZScore IS NULL THEN NULL
        ELSE 1.0 - (0.5 * (1.0 + ERF(z.ZScore / SQRT(2.0))))
       END AS Probability,

    CASE
        WHEN z.ZScore IS NULL THEN NULL
        WHEN z.ZScore >= 3.5 THEN 'ANOMALY'
        WHEN z.ZScore >= 3.0 THEN 'VERY HIGH'
        WHEN z.ZScore >= 2.0 THEN 'HIGH'
        ELSE 'NORMAL'
    END AS AnomalyClass
FROM zcalc z;
```