

Case Study Brief - Data Analyst 2025 rev 1.1

Focus: Success Pattern Discovery • SQL Logic Design • AI-Powered Talent Matching • Business Storytelling

Overview

Company X is developing a **Talent Match Intelligence** system to help leaders identify what makes top-performing employees successful and to find individuals who share those characteristics for **succession**.

In this case, you’ll simulate the real data-analysis workflow behind that system.

You will:

1. **Discover** what drives employee success.
2. **Formalize** those drivers into a clear, explainable **Success Formula**.
3. **Translate** that logic into **SQL** that computes match scores.
4. **Present** your findings through a lightweight **AI-powered app** and **dashboard** that generates job profiles and visual insights.

▲ Important Note:

In real-world analytics, you rarely start with complete knowledge. This case study is designed to test your ability to **learn, explore, and adapt**. If you don’t have an HR or psychology background, you’re expected to research and understand key terms and metrics yourself. Saying “I’m not familiar” is **not an acceptable limitation**. Discovering what you don’t know and translating it into actionable analysis **is the job itself**.

Tools You’ll Need

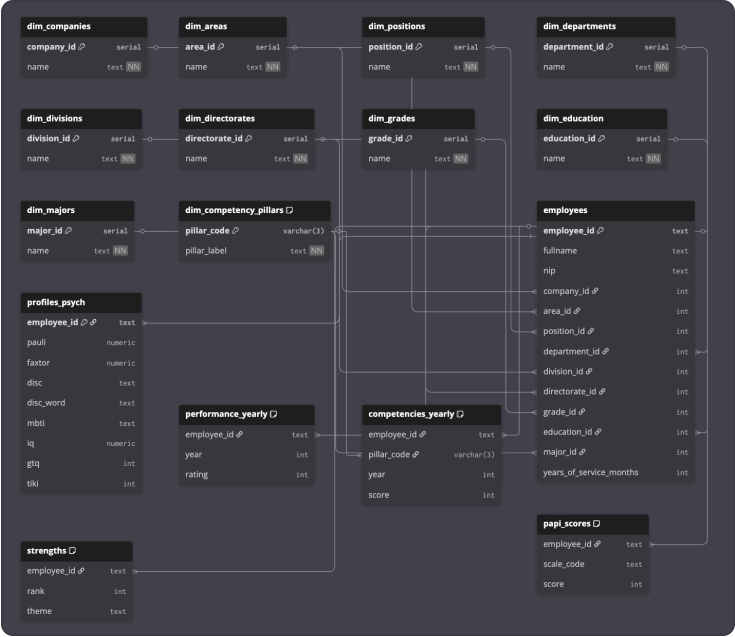
Category	Tool / Platform	Notes
Database	Supabase (Postgres)	For data storage, queries, and SQL logic
Programming & Analysis	Python / R / SQL	For analysis, queries, and formula exploration
Visualization & App	Streamlit, or any enterprise-grade dashboard framework	For insight dashboards and job vacancy visualizations
AI Model	OpenRouter (or any free LLM)	To generate job requirements, descriptions, and competency lists
Version Control	GitHub	For sharing scripts and documentation
Report Format	PDF (Case Study Report)	Final submission format (see template)

Dataset & ERD

You’ll use the dataset

Study Case DA

, Glossary Data, modeled using the [ERD](#) provided below.



Key Entities

Category	Main Tables	Description
Employee & Structure	employees, dim_companies, dim_directorates, dim_positions, dim_grades, etc.	Organizational context and employee profile.
Performance	performance_yearly	Annual rating (1–5). High performers = rating 5.
Assessments	profiles_psych, papi_scores, strengths	Cognitive, psychometric, and behavioral data.
Competencies	competencies_yearly, dim_competency_pillars	Yearly competency scores and soft skills.

This schema provides complete coverage of behavioral, cognitive, and organizational attributes.

Case Study Flow - The Red Thread

Step 1 - Discover the Pattern of Success

Your first objective is to identify *why* some employees achieve **rating 5** while others don't. Perform data exploration and visualization to uncover what differentiates high performers across:

- Competency pillars (competencies_yearly + dim_competency_pillars)
- Psychometric profiles (papi_scores, profiles_psych)
- Behavioral data (strengths)
- Contextual factors (grade, years_of_service_months, education, etc.)

Use storytelling visuals (heatmaps, radar charts, correlation plots, comparison matrices) to explain the **why**, not just the **what**.

Then, synthesize those findings into a **Success Formula**, a weighted structure that captures how performance emerges from multiple factors.

Note: Any example like

SuccessScore = 0.3*Cognitive + 0.2*Leadership + ...

is **oversimplified**.

The real challenge is to balance *many interacting variables*, across competencies, assessments, and traits, into one explainable success framework. At this point, you are allowed to create **formulas using rule-based logic**.

Deliverable:

- Final **Success Formula** with justification.
- Supporting analysis and visuals demonstrating how it was derived.

Understanding Talent Group Variables (TGV) and Talent Variables (TV)

Before moving into the SQL logic, it's important to understand how company X conceptualizes the building blocks of success.

- **Talent Group Variables (TGV)** represent **broad skill or behavioral categories** that affect performance.
Examples: *Leadership, Cognitive Ability, Personality, Teamwork, Technical Expertise*.
- **Talent Variables (TV)** are **specific measurable components** within each TGV.
Examples:
 - Under *Cognitive Ability*: *IQ Score, Numerical Reasoning, Problem Solving*
 - Under *Leadership*: *Decision Making, Strategic Thinking, Accountability*

Every TV contributes to its parent TGV.

In your SQL logic, you will:

1. Compare each **TV** between the candidate and benchmark (TV match rate).
2. Aggregate those into **TGV match rates** (group-level averages).
3. Combine all TGVs into a **Final Match Rate** (overall fit score).

In short:

TV = individual measurable variable

TGV = category grouping of related TVs

Both are essential for interpreting talent alignment.

You can check the detail **Talent Group Variables (TGV) and Talent Variables (TV)** [here](#).

Step 2 - Operationalize the Logic in SQL

Managers pick one or more **talent benchmarks** (rating = 5) to define an ideal profile for a vacancy.

Your SQL must calculate how closely every employee matches that benchmark.

talent_benchmarks table

Column	Description
job_vacancy_id	Unique role ID
role_name	Job title
job_level	Level / grade
role_purpose	1-2 sentence summary
selected_talent_ids	Array of benchmark employee IDs
weights_config	Custom weights per TV or TGV

Weights may be **equal or custom** at both TV and TGV levels.

Matching Algorithm

Use modular CTEs to implement the logic below.

1. **Baseline Aggregation (on the fly)**
 - a. For each TV, compute a **benchmark baseline** (median of selected talent scores).
2. **TV Match Rate (Employee × TV)**
 - a. For **numeric variables**:
Compare the employee's score directly against the baseline.
Example:
 - i. If benchmark IQ = 110 and candidate IQ = 100 → match rate = 100 / 110 = 90.9 %.
 - ii. If scoring direction is “lower is better”, invert the ratio: $((2 * \text{benchmark_score} - \text{user_score}) / \text{benchmark_score}) * 100$.
 - b. For **non-numeric / categorical variables**:
Use boolean comparison, i.e., exact match = 100 %, no match = 0 %.
3. **TGV Match Rate (Employee × TGV)**
 - a. Average the TV match rates within each TGV.
 - b. Apply equal or custom TV weights if provided.
4. **Final Match Rate (Employee)**

- a. Weighted average across all TGVs.
- b. Apply equal or custom weights from `weights_config`.

Expected SQL Output Columns

Column	Meaning
<code>employee_id</code>	Candidate ID
<code>directorate</code>	Directorate
<code>role</code>	Position title
<code>grade</code>	Grade / level
<code>tgv_name</code>	Talent Group Variable (e.g., Cognitive)
<code>tv_name</code>	Talent Variable (e.g., Strategic Thinking)
<code>baseline_score</code>	Benchmark average for this TV
<code>user_score</code>	Candidate score for this TV
<code>tv_match_rate</code>	Match % for this TV
<code>tgv_match_rate</code>	Avg/weighted match within TGV
<code>final_match_rate</code>	Weighted overall match %

⚠ Important Note:

Deliver → a well-documented SQL script producing this table with comments explaining each step.

The query results must be displayed.

Step 3 - Build the AI Talent App & Dashboard

Your final step is to turn the SQL results into actionable insight through an a parameterized, **AI-powered interface** and **visual dashboard**. This app must handle any new input at runtime, not a static or pre-baked dashboard.

The goal is to demonstrate how you can **process, visualize, and narrate data** to bring clarity, **not** to build a production-grade web app.

⚠ Important Note:

The focus is **not on app development or UI polish**.

We’re evaluating your ability to **transform complex data into meaningful visuals and business insights**.

Think like an *analyst who can code for clarity*, not a software engineer

Your app must be deployed publicly so that we can easily access and review it.

Inputs (runtime, user-provided)

- Role name
- Job level
- Role purpose
- Selected benchmark employee IDs

When the user submits new inputs, your logic must:

1. Record or parameterize a new `job_vacancy_id` inside `talent_benchmarks`,
2. Recompute baselines dynamically from the selected benchmark employees,
3. Re-run your parameterized SQL query, and
4. Regenerate the profile, ranking, and visuals without editing the code.

Example:

Column	Desc
Role Name	Data Analyst
Job Level	Middle
Role	Data Analyst
Selected benchmark employee IDs	312, 335, 175

1. Role Information

Role Name ⓘ

Ex. Marketing Manager

Job Level

Choose your job level

Role Purpose

1-2 sentences to describe role outcome

Example: Ensure production targets are met with optimal quality and cost efficiency

Employee Benchmarking

Select Employee Benchmarking (max 3)

Choose employee benchmarking

Generate Job Description & Variable score

Outputs

1. AI-Generated Job Profile

- Job requirements, description, and key competencies (generated via any LLM, e.g., OpenRouter).



Column	Desc
--------	------

Job requirements	<p>SQL expertise: complex joins, window functions, CTEs, performance tuning basics.</p> <p>R or Python for analysis (pandas/dplyr), statistics, and quick prototyping (Streamlit/Shiny/Dash).</p> <p>BI tooling: Looker/Power BI/Tableau (modeling, permissions, and production dashboards).</p> <p>Data modeling fundamentals (star schema, slowly changing dims, metrics layer) and version control.</p> <p>Visualization best practices and data storytelling for non-technical audiences.</p> <p>Strong analytical thinking: hypothesis framing, causal caveats, sensitivity checks.</p> <p>Bias & fallacy awareness: sampling bias, survivorship, p-hacking, confirmation bias; practices to mitigate.</p> <p>Communication in English & Bahasa; stakeholder management across levels.</p>
Job description	You turn business questions into data-driven answers. You'll own the analysis cycle end-to-end: understand context, shape clear dashboards, and craft narratives that drive decisions. You balance technical depth (SQL, R/Python, BI) with business rigorous thinking , and bias-aware judgement.
key competencies	<p>SQL (Postgres/Snowflake/BigQuery), Git, DBT (nice), Airflow (nice)</p> <p>R/Python (pandas/numpy/scipy or tidyverse), Streamlit/Shiny</p> <p>Looker/Tableau/Power BI, Excel/Sheets for quick checks</p>

1. **Ranked Talent List**

- Display the output of your SQL logic (at minimum): employee_id, name, final_match_rate, and supporting fields (e.g., top TGVs/TVs, strengths, gaps)

2. **Dashboard Visualization**

- Provide clear, interactive visuals for each new input/job vacancy:
- Match-rate distributions
- Top strengths and gaps across TGVs
- Benchmark vs candidate comparisons (radar, heatmap, bar plots)
- Summary insights explaining why certain employees rank highest
-  The dashboard should let stakeholders see and understand the data.
-  It should not focus on UI frameworks, routing, or authentication.

Deliverable:

- Working app/dashboard that connects your SQL logic, AI generation, and visual insight.

Example

Job Details

All fields below are required. Please add at least one item for each category.

Key Responsibilities

Add an item

Add

•

Example placeholder text

•

Example placeholder text

•

More example placeholder text

•

Placeholder for solution

Work Inputs

Add an item

Add

Work Outputs

Add an item

Add

Qualifications

Add an item

Add

•

Experience placeholder

•

Demom placeholder

•

Experience placeholder

Competencies

Add an item

Add

•

JavaScript Frameworks

•

UI/UX Principles

•

Problem Solving

•

Team Collaboration

•

Code Optimization

•

Agile Methodologies

•

Adaptability

•

Innovative Thinking

Q Cari nama talent, contoh: john doe

0

Move Employee to

	NAME	MATCH RATE	Employee Information				
			Role	Division	Department	Directorate	Job_level
<input type="checkbox"/>		87%	F				
<input type="checkbox"/>		86%					
<input type="checkbox"/>		86%					
<input type="checkbox"/>		86%	Y				
<input type="checkbox"/>		86%					
<input type="checkbox"/>		86%					
<input type="checkbox"/>		86%	E				
<input type="checkbox"/>		86%	V				
<input type="checkbox"/>		86%	E				
<input type="checkbox"/>		86%					

Showing 1 to 10 of 340 results

< Previous

1

2

3

...

34

Next >

Glossary

Assessment Context

PAPI Kostick (Work Preferences)

- **What:** Work-style preferences (initiative, leadership, conformity, etc.).
- **In data:** `papi_scores` (scales `Papi_N ... Papi_W`, 1–9).
- **Note:** Some scales are **inverse** in the match logic (watch the Z/K-style inversions).

MBTI (Type Preference)

- **What:** Four dichotomies (e.g., E/I, S/N, T/F, J/P) for preference, not ability.
- **In data:** `profiles_psych.mbti` (text; may be messy casing/spaces).
- **Note:** Clean to 16 valid types if needed.

DISC (Behavioral Style)

- **What:** Dominance, Influence, Steadiness, Conscientiousness preferences.
- **In data:** `DISC`, `first_char/second_char`, `first_word/second_word`, `DISC_word`.
- **Note:** Useful for narrative fit.

IQ / Cognitive Index

- **What:** General cognitive/problem-solving potential (proxy).
- **In data:** `profiles_psych.iq` (approx. 80–140).
- **Note:** If included, treat extremes/outliers carefully (bias risk).

GTQ (Aptitude Subtests)

- **What:** Short cognitive/aptitude components (e.g., reasoning, numeracy).
- **In data:** `GTQ1-GTQ5` (1–10), `GTQ_total`.

TIKI (Short Cognitive/Attention Tasks)

- **What:** Brief attention/processing subtests.
- **In data:** `Tiki1-Tiki4` (1–10).

Pauli (Kraepelin-type Mental Arithmetic)

- **What:** Continuous addition task, **speed + accuracy + mental stamina**.
- **In data:** `profiles_psych.pauli` (numeric 20–100 in this dummy).

Faxtor (Internal Cognitive/Attention Index)

- **What:** Internal composite (attention/processing) used in the dummy.
- **In data:** `profiles_psych.faxtor` (numeric 20–100).

CliftonStrengths (Top Themes)

- **What:** Talent themes (e.g., Achiever, Strategic, Learner) reflecting natural patterns.
- **In data:** `strengths` with **rank 1–14** (theme names).

Data Dictionary

1) Core Dimensions

1.1 `dim_companies`

Purpose: Organization entity.

- `company_id` · `int` · PK · surrogate key.
- `name` · `text` · **Unique, NOT NULL**.

1.2 `dim_areas`

Purpose: Work area / site.

- `area_id` · `int` · PK.
- `name` · `text` ·

1.3 `dim_positions`

Purpose: Role family / position.

- `position_id` · `int` · PK.
- `name` · `text` · e.g., “Data Analyst”, “Brand Manager”. **Unique.**

1.4 `dim_departments`

Purpose: Department.

- `department_id` · `int` · PK.
- `name` · `text` · e.g., “Marketing”, “HR”, “IT”. **Unique.**

1.5 `dim_divisions`

Purpose: Division.

- `division_id` · `int` · PK.
- `name` · `text` · division label.

1.6 `dim_directorates`

Purpose: Directorate.

- `directorate_id` · `int` · PK.
- `name` · `text` · e.g., “HR & Corporate Affairs”, “Commercial”.

1.7 `dim_grades`

Purpose: Job grade/band.

- `grade_id` · `int` · PK.
- `name` · `text` · e.g., “III”, “IV”, “V”.

1.8 `dim_education`

Purpose: Highest education.

- `education_id` · `int` · PK.
- `name` · `text` · e.g., “SMA”, “D3”, “S1”, “S2”.

1.9 `dim_majors`

Purpose: Field of study.

- `major_id` · `int` · PK.
- `name` · `text` · e.g., “Psychology”, “Engineering”.

1.10 `dim_competency_pillars` (*Historical competency pillars*)

Purpose: Pillars used in historical competency ratings.

- `pillar_code` · `varchar(3)` · PK · e.g., `GDR`, `CEX`, `IDS`, `QDD`, `STO`, `SEA`, `VCU`, `LIE`, `FTC`, `CSI`.
- `pillar_label` · `text` · human-readable label.

2) Core Entities / Facts

2.1 `employees`

Purpose: Person master.

- `employee_id` · `text` · PK · unique internal ID.
- `fullname` · `text`.
- `nik_baru` · `text`.
- `company_id` · `int` · FK → `dim_companies`.
- `area_id` · `int` · FK → `dim_areas`.
- `position_id` · `int` · FK → `dim_positions`.
- `department_id` · `int` · FK → `dim_departments`.
- `division_id` · `int` · FK → `dim_divisions`.
- `directorate_id` · `int` · FK → `dim_directorates`.

- `grade_id` · `int` · FK → `dim_grades`.
- `education_id` · `int` · FK → `dim_education`.
- `major_id` · `int` · FK → `dim_majors`.
- `years_of_service_months` · `int` · 0–180 typical.

2.2 `profiles_psych` (one row per employee)

Purpose: Non-yearly psychometric profile & derived fields.

- `employee_id` · `text` · PK, FK → `employees`.
- `pauli` · `numeric` · ~20–100 (dummy).
- `factor` · `numeric` · ~20–100 (dummy).
- `disc` · `text` · e.g., “DI”, “SC” (messy variants possible).
- `first_char`, `second_char` · `text` · D/I/S/C.
- `first_word`, `second_word`, `disc_word` · `text`.
- `enneagram` · `int` · 1–9.
- `mbti` · `text` · 16 types (messy casing/spaces may occur).
- `iq` · `numeric` · ~80–140 (dept-level missingness possible).
- `gtq1..gtq5` · `int` · 1–10.
- `gtq_total` · `int`.
- `tiki1..tiki4` · `int` · 1–10.

2.3 `papi_scores`

Purpose: PAPI Kostick scales (work preferences).

- `employee_id` · `text` · FK.
- `scale_code` · `text` · in { `Papi_N`, `Papi_G`, `Papi_A`, `Papi_L`, `Papi_P`, `Papi_I`, `Papi_T`, `Papi_V`, `Papi_X`, `Papi_S`, `Papi_B`, `Papi_O`, `Papi_R`, `Papi_D`, `Papi_C`, `Papi_Z`, `Papi_E`, `Papi_K`, `Papi_F`, `Papi_W` } (set may vary by form).
- `score` · `int` · **1–9**.
PK: (`employee_id`, `scale_code`)

2.4 `strengths`

Purpose: CliftonStrengths themes (Top 14).

- `employee_id` · `text` · FK.
- `rank` · `int` · **1..14** (unique per employee).
- `theme` · `text` · from Gallup’s 34 themes (e.g., Achiever, Strategic, Learner, Relator, etc.).
PK: (`employee_id`, `rank`)

2.5 `performance_yearly`

Purpose: Annual performance.

- `employee_id` · `text` · FK.
- `year` · `int` · 2021–2025.
- `rating` · `int` · **1–5** (some outliers injected).
PK: (`employee_id`, `year`)

2.6 `competencies_yearly`

Purpose: Historical competency ratings (10-pillar model).

- `employee_id` · `text` · FK.
- `pillar_code` · `varchar(3)` · FK → `dim_competency_pillars`.
- `year` · `int` · 2021–2025.
- `score` · `int` · **1–5**.
PK: (`employee_id`, `pillar_code`, `year`)

Note: This is **not** the new TGV framework; it’s historical context.

2.7 `employee_archetypes`

Purpose: Top-3 archetypes per employee (from flags).

- `employee_id` · `text` · FK.
 - `archetype_code` · `text` · FK → `dim_archetypes`.
- PK:** (`employee_id`, `archetype_code`)

Submission Package

Please compile your submission into a **single Case Study Report (PDF)** following the structure below.

Candidate Information

- **Full Name:**
- **Email Address:**

Repository Link

- Provide a direct link to your GitHub repository containing all source files.
- **▲ Important:** Do **not** use the word “**Rakamin**” anywhere in your repository name, commits, or documentation. This is to reduce plagiarism risk.
- Example:
`github.com/username/talent-match-intelligence`

Your repository must include:

- SQL scripts
- App code and dashboard
- `README.md` with setup instructions
- Any supporting assets (datasets, configuration files, etc.)

Main Report

Your report must be clear, concise, and business-ready. Written and formatted to enterprise or industry presentation standards (e.g., consulting-grade deck or executive report).

Use the following structure:

1. **Executive Summary**
 - a. Project overview, objectives, key outcomes, and impact
2. **Success Pattern Discovery (Deliverable #1)**
 - a. Enterprise-standard report/deck presenting your analysis process, findings, and key insights
 - b. Visuals, charts, and narrative that would be understandable to non-technical stakeholders
 - c. Final **Success Formula** and rationale
3. **SQL Logic & Algorithm (Deliverable #2)**
 - a. Explanation of your SQL approach
 - b. Query structure and CTE logic overview
 - c. Snapshot of the output table (sample rows and key columns)
4. **AI App & Dashboard Overview**
 - a. **Deployment Link.** Provide a **publicly accessible link** to your deployed output for **step 3**.
 - b. Inputs and outputs of the AI component
 - c. Key visualizations and insight narratives
 - d. Example screenshots of your dynamic dashboard
5. **Conclusion**
 - a. Reflections, challenges faced, and ideas for improvement

Additional Files (if any)

Attach or link to:

- Analysis notebooks (Python, Jupyter, etc.)
- Generated visuals or supporting documentation