# CASE STUDY: AI-Powered Org Intelligence Assistant for Beta Corp

## Project Background

Client: Beta Corp (Fictional mid-sized enterprise)

Employees: ~15,000 across global offices

Beta Corp is undergoing rapid growth, and leadership suspects inefficiencies in the organization’s structure — such as unbalanced team sizes, excessive managerial span of control, and inconsistent supervisory layers. At the same time, HR is frequently swamped with repetitive internal data queries from managers and department heads.

The leadership team is looking to understand:

1. How the company is structured — across locations, departments, and management layers
2. What inefficiencies or red flags exist in the org design
3. How an AI assistant could help scale HR analytics and reduce manual effort

As an lead consultant, you're expected to:

1. Conduct analysis on the HRIS dataset provided
2. Build a working prototype of an internal HR Assistant that can answer data-related questions via natural language
3. Deliver your results in a user-friendly app

## Objectives

You are expected to deliver the following in 4 days:

### PART 1: Exploratory Data Analysis (EDA)

You are provided with a dataset employee\_data\_15000.csv with the following fields:

|  |  |
| --- | --- |
| **Column Name** | **Definition** |
| employee\_id | Unique numeric identifier assigned to each employee. |
| first\_name | Employee’s first name. |
| last\_name | Employee’s last name. |
| department | The functional business unit the employee belongs to (e.g., Engineering, Sales). |
| job\_title | The full designation of the employee’s role (e.g., Software Engineer, Sales Executive). |
| job\_level | The seniority or grade level of the job, usually numeric (e.g., 1 = entry-level, 5 = senior/executive). |
| location | The physical office location where the employee is based (e.g., New York, Chicago). |
| region | The broader geographical region the location falls under (e.g., NA = North America, EMEA, APAC). |
| manager\_id | Employee ID of the direct (line) manager the employee reports to. |
| joining\_date | Date when the employee officially joined the organization. |
| exit\_date | Date when the employee exited the organization (empty if still active). |
| performance\_rating | Most recent performance score of the employee, typically on a numeric scale (e.g., 1–5). |
| tenure\_years | Number of years the employee has been with the company (calculated from joining\_date to today or exit\_date). |
| is\_active | Indicates whether the employee is currently active in the organization (TRUE/FALSE). |
| supervisor\_id | Employee ID of a secondary/dotted-line supervisor (used in matrix organizations for cross-functional oversight). |

**Tasks:**

1. **Clean and preprocess** the data:
   * Parse joining\_date, exit\_date as datetime
   * Derive tenure\_years if not present or recalculate
   * Handle missing or inconsistent manager\_id, supervisor\_id
   * Standardize job\_title, location, and department names (if needed)
2. **Perform EDA** to explore:
   * Employee distribution by department, location, job level
   * Span of control for each manager (number of direct reports)
   * Depth of hierarchy / organizational layers
   * Geo-distribution of employees
   * Attrition trends if is\_active is False
3. **Deliver 2–3 insights** that may indicate:
   * Organizational inefficiencies
   * Overburdened managers or isolated roles
   * Uneven headcount or supervisory relationships
4. **Visualizations (at least 2):**
   * Department-wise headcount
   * Histogram of span of control or team size distribution
   * Optional: heatmap of regions vs job levels, org depth metrics

### PART 2: Build a Data Agent for Natural Language HR Queries

**Goal:**

Enable a GenAI assistant to answer **natural language questions** about the employee data by querying a local database (e.g., SQLite) using SQL or Pandas logic.

**Requirements:**

* Convert the CSV into a **SQL database** (e.g., SQLite) or use a Pandas DataFrame
* Build an **Agent using LangChain** (or equivalent framework) that can:
  + Parse natural language questions
  + Generate SQL queries (or Pandas logic)
  + Retrieve and present answers using actual HRIS data

**Sample Queries the Assistant Should Handle:**

* "How many employees are in the Engineering department in New York?"
* "Which manager has the highest number of direct reports?"
* "What is the average tenure in the Sales department?"
* "Who are the top 3 longest-tenured employees in Marketing?"
* "List teams reporting to the Director of Operations in APAC"

**Expectations:**

* Responses must be **data-grounded**, not hallucinated
* Some error handling (e.g., “I don’t have that info”) is ideal
* Use LangChain, OpenAI, or open-source LLMs

### PART 3: Streamlit App Prototype ((Bonus – Optional but Recommended)

Build a lightweight web app with two tabs:

**Tab 1: Org Insights Dashboard**

* Display visuals and summary insights from your EDA (Part 1)
* Include at least 2 charts and textual commentary

**Tab 2: Ask the HR Assistant**

* Input box for user queries (about HR data)
* AI-generated output via your data agent
* Include basic UI features:
  + Loading message
  + Scrollable response box or formatting

### Submission Guidelines

Organize your submission as follows:

├── eda/

│ └── your\_notebooks\_or\_scripts.ipynb

├── app/

│ └── streamlit\_app.py

├── agent/

│ └── langchain\_agent.py (or similar)

├── data/

│ └── employee\_data\_15000.csv

├── requirements.txt

├── final\_report.pdf OR final\_presentation.pdf

## Final Report Expectations

Keep this short (max 3–5 pages or 7–10 slides), but include:

* **Executive Summary**: What you did and why
* **Org Insights**: Summary of EDA findings + visuals
* **Agent Design**: Brief overview of how the AI agent works (flow of question → SQL → response)
* **Q&A Examples**: At least 3 examples with sample inputs and answers
* **Cloud Architecture**: (Optional) Diagram + deployment plan
* **Recommendations**: Any proposed next steps for Beta Corp

|  |  |
| --- | --- |
| **Area** | **What We’re Looking For** |
| EDA + Insights | Clean and scalable code, thoughtful insights, and strong use of visualizations |
| Agentic AI Logic | Good tool choice (LangChain, LLMs), accurate SQL generation, grounded responses |
| App UX | Simple, clear UI with loading states, intuitive layout |
| Problem Solving | Clear breakdown of solution, smart tradeoffs |
| Communication | Well-structured report/slides, clear explanation of your work |