**MrBeast Senior HRIS Engineer Take-Home Project**

**Overview**

Welcome! This take-home project is designed to simulate the technical challenges you’ll face as a Senior HRIS Engineer at MrBeast. You’ll work with mock HR data, build integrations, transform data, and visualize insights—mirroring the real-world responsibilities of this role.

**Project Scenario**

You are simulating the integration of data from two HR systems: Recruiting and Payroll. Your goal is to build a pipeline that extracts, transforms, and loads this data into a centralized SQL database, then creates a REST API and visualizes insights for the People Ops and Leadership teams.

**Provided Files**

You have been provided with a CSV file with three worksheets: Employees, Applicants, employment type

* [HRIS TAKE HOME PROJECT\_DATA](https://docs.google.com/spreadsheets/d/1_JFsDUKG5CqZmvaiIgq6or4plFQddcM0gRB5Kqi2Q70/edit?gid=0#gid=0)

These simulate exports from separate HR systems. The data may include small inconsistencies to reflect real-world conditions.

**Your Tasks**

**1. Load the Data into a Local SQL Database**

* Choose any SQL database (PostgreSQL or SQLite preferred).
* Use a preferred script to load each sheet as a separate table.
* Normalize the schema and apply appropriate data types.

**2. Data Pipeline & Transformation (DBT/SQL)**

* Use DBT or SQL to:
  + Clean and standardize the data (e.g., date formats, missing values).
  + Create a view or model to calculate "Time-to-Hire"

**3. Build a REST API to Serve Cleaned Data**

* Using the language and framework of your choice
* Create at least **two endpoints**, for example:
  + GET /hiring-metrics — returns average time-to-hire by department
  + GET /applicants/status-summary — returns applicant counts by status
* Ensure endpoints:
* Return data in clean, paginated JSON
* Follow REST conventions
* Handle errors appropriately (e.g., 404s, 500s)

**4. Automation (Cron)**

* Schedule a daily run at 2 AM using a cron expression and/or shell script.

**5. Data Visualization**

* Create static charts or a lightweight dashboard that displays:
  + Count of applicants by status (e.g., hired, rejected)
  + Average time-to-hire by department
  + Any additional metrics you believe a CHRO or CPO would find valuable

Use any tool you’re comfortable with: Python (Plotly, Matplotlib), Tableau, Metabase, etc.

**6. Documentation**

* Write a clear README that covers:
  + Step-by-step setup instructions
  + Design decisions (e.g., schema structure, transformation logic)
  + Any assumptions or known limitations
  + Your thought process on what metrics are useful to leadership

**Submission Instructions**

* Upload your work to a GitHub repository that includes:
  + All source code (API, data loading, SQL/DBT, visualizations)
  + Your README
  + Exported dashboard or charts (if applicable).
  + Share the repository with [Nagesh1011](https://github.com/Nagesh1011), [Bymc1978](https://github.com/Bymc1978), [TuxGamer](https://github.com/TuxGamer), [cardonal96](https://github.com/cardonanl96)
* Do not include the original provided data in your submission.

Email [Byronm@mrbeastyoutube.com](mailto:Byronm@mrbeastyoutube.com), [nagesh@mrbeastyoutube.com](mailto:nagesh@mrbeastyoutube.com) and [Tux@mrbeastyoutube.com](mailto:Tux@mrbeastyoutube.com) once completed. Include any links and attachments relevant to your project.

**Evaluation Criteria**

|  |  |
| --- | --- |
| **Area** | **What We’re Looking For** |
| API Design | RESTful standards, error handling, scalability |
| Code | Readability, reproducibility, structure |
| DBT/SQL | Data model clarity, schema design, transformation quality |
| Automation | Cron setup, logging, reliability |
| Visualization | Relevant, clear, insightful metrics for business stakeholders |
| Documentation | Reproducibility, clarity, assumptions explained |

**Bonus Points**

* Add unit tests for API or transformation logic
* Implement basic auth for your APIs
* Containerize your solution using Docker.
* Add basic alerting (e.g., email/Slack on pipeline failure).

**Notes**

* Focus on clean, maintainable code and realistic data handling.
* Feel free to go beyond what's asked — add metrics, improve data quality, or structure your API for real-world use.
  + Your approach matters as much as your output.
* If you hit a blocker, please reach out to [byronm@mrbeastyoutube.com](mailto:byronm@mrbeastyoutube.com).
  + Since this is a senior role, we encourage you to proceed using your best judgment and note any open questions or assumptions in your README.

Thank you for your interest in MrBeast’s People Tech team. We look forward to reviewing your work!

✅ **MrBeast Senior HRIS Engineer Take-Home Checklist**

| **Task Area** | **Action Item** | **Status** | **Notes** |
| --- | --- | --- | --- |
| **1. Data Loading** | Use SQLite for local DB | ✅ | SQLite preferred for speed/prototyping |
|  | Load each sheet into a table | ✅ | Tables: employees, applicants, employment\_types |
|  | Normalize schema and apply data types | ✅ | Convert date columns, set joins via email/type\_id |
|  | Handle duplicates if present | ⚠️ | Consider .drop\_duplicates() in data\_loader.py |
| **2. Data Pipeline & Transformation** | Clean and standardize data | ✅ | Date formats, missing values handled |
|  | Calculate “Time-to-Hire” metric | ✅ | In API SQL or via a time\_to\_hire view |
|  | Use DBT optionally (not required here) | ❌ | Using raw SQL is fine |
| **3. REST API** | Flask app with endpoints | ✅ | Defined in app.py |
|  | GET /hiring-metrics returns avg time-to-hire | ✅ | Aggregated by department |
|  | GET /applicants/status-summary | 🟡 | Still needs to be added |
|  | Return clean JSON, paginated (if applicable) | ⚠️ | Add query params support |
|  | Include error handling (404, 500) | ⚠️ | Wrap with try/except blocks |
| **4. Automation (Cron)** | Write shell script or scheduler | ❌ | scheduler.sh to trigger data\_loader.py |
|  | Schedule daily at 2AM | ❌ | 0 2 \* \* \* python3 path/to/script.py |
| **5. Visualization** | Plot count of applicants by status | ❌ | Plotly/Matplotlib suggested |
|  | Plot avg time-to-hire by dept | ❌ | Bar chart ideal |
|  | Add additional relevant metrics | ❌ | Turnover, bottlenecks, role-based variance? |
| **6. Documentation** | README with setup and structure | 🟡 | Include install, run, config steps |
|  | Explain transformations and metrics | 🟡 | Make assumptions explicit |
|  | Include known limitations | 🟡 | Time zone issues? Small sample size? |
|  | Link GitHub repo and share with contacts | ❌ | Must include all four GitHub handles |
|  | Email completion to listed recipients | ❌ | Attach repo URL and charts (no original data) |

Data relationship between tables:

 📎 **Employees ↔ Employment Types**: Row counts match, so linking by ID or sequence indexing is solid.

* 🔍 **Applicants ↔ Employees**: 164 matched names confirm real-world hiring transitions.
* ⚠️ **Applicants IDs** ≠ Employee IDs: They’re independently assigned, so name normalization is your bridge.
* 🔗 How test\_views.py Interconnects

| **Script** | **Role** | **Dependency Chain** |
| --- | --- | --- |
| data\_loader.py | Cleans + loads Excel data | Creates hris\_project.db |
| transform.py | Defines SQL views | Depends on DB from loader |
| app.py | Serves endpoints via Flask | Depends on views in DB |
| test\_views.py | Validates views + logic | Depends on view definitions |