Design an AI agent that automates data engineering tasks by interpreting product backlog items, generating ETL pipelines, applying transformations, and delivering analytical outputs (e.g., scoring via a model).
It’s required to consider correct interpretation of backlog items, optimal code generation (e.g., partition handling), data quality checks and error handling, and minimal manual intervention in the pipeline.

Key Requirements

Backlog Interpretation

Parse natural language backlog items to extract requirements (source tables, transformations, business rules).
Use LangChain + AutoGen for NLP task breakdown.

Code Generation

Auto-generate Spark SQL/PySpark for ETL.
Suggest joins, filters, and aggregations.
Create dbt models or Airflow DAGs if applicable.

Data Quality & Validation

Integrate Great Expectations for automated checks.
Flag anomalies with human-in-the-loop approval.

Model Integration

Apply an AutoML-generated scoring model.
Output results with metadata (confidence scores, validation logs).

Deployment & Version Control

Auto-push code to Git or orchestrate via Airflow.
Include human approval hooks before production.

Possible Technology Stack

NLP & Agents: GPT-4/Claude + LangChain + AutoGen
ETL: PySpark, dbt, Airflow
Validation: Great Expectations
AutoML: H2O, PyCaret, or custom MLflow pipelines
RLHF: Fine-tuning based on human feedback

Submission

GitHub repo with agent code, sample backlog items, and demo outputs.
Short report explaining design choices and limitations.

Hint:

- Focus on modularity - break tasks into sub-agents (e.g., SQL generator, validator). Use rule-based fallbacks for cost/LLM limitations.
- Dataset to refer while exercising the implementation is proved, problem statement for model/engine – scoring that that range from 0 to 100 to each applicant is of repaying a loan?