Excellent. Having a solid grasp of your existing skills makes choosing the right technologies much more straightforward. Your current toolkit is fantastic and aligns perfectly with building a high-impact, end-to-end project.

Let's design a modern, impressive, and highly practical tech stack for your project that leverages what you know and adds a few key technologies that will make your resume stand out.

**The "Right Tool for the Job" Philosophy**

For this project, we will intentionally select tools that are not just impressive, but are also the *appropriate* choice for the scale of this problem. For example, while you know PySpark, using it for a questionnaire dataset (which is likely thousands, not billions, of rows) would be overkill. Sticking with Pandas demonstrates good technical judgment.

Here is a breakdown of the recommended technologies for each phase of the project.

**Phase 1: Data Engineering & The ETL Pipeline**

**Goal:** Build a robust, automated pipeline to process the raw questionnaire data and load it into a proper database.

* **Database: PostgreSQL**
  + **Why:** You already know it. It's a powerful, professional-grade database that is far more impressive on a resume than SQLite or a flat file.
  + **Implementation:** Instead of managing it on EC2, I strongly recommend using a free-tier cloud Postgres provider like **Supabase** or **Neon**.
    - **Supabase:** This is my top recommendation. It's PostgreSQL with a user-friendly interface and a generous free tier. Listing "Supabase" on your resume is a modern keyword that signals you know current tools. It's incredibly easy to set up.
    - **Neon:** A great serverless Postgres alternative.
* **ETL Scripting: Python + Pandas**
  + **Why:** This is the industry standard for this scale of data manipulation. Your existing skills are perfect here.
  + **Implementation:** Create a main Python script (etl.py) that handles the entire Extract, Transform, and Load process. Use libraries like psycopg2-binary or sqlalchemy to connect your Python script to your Postgres database (Supabase provides the connection string directly).
* **Secret Sauce (Advanced): LLM for Data Cleaning**
  + **Why:** To showcase your LangChain/Ollama skills in a practical engineering context. You can use a local LLM to help categorize messy, user-submitted technology names that your rule-based system might miss.
  + **Implementation:**
    1. In your ETL script, for any technology name that doesn't match your predefined categories, make an API call to a local **Ollama** model (like Llama 3).
    2. Use a prompt like: "Given the technology name '{tech\_name}', which of the following categories does it best fit into? [Frontend Framework, Backend Framework, Cloud Platform, ...]. Respond with only the category name."
    3. This is an *extremely* impressive and modern data engineering technique to show an interviewer.

**Phase 2: Analytics & AI Modeling**

**Goal:** Analyze the clean data, build a predictive model, and present everything in an interactive application.

* **Web Application / Dashboard: Streamlit**
  + **Why:** You know it, it's pure Python, and it's perfect for creating beautiful data apps with minimal effort. You can build both your analytics dashboard and your AI model interface in one place.
  + **Implementation:** Structure a multi-page Streamlit app. Page one can be the "Analytics Dashboard," and page two can be the "AI Predictions."
* **Machine Learning Model: scikit-learn**
  + **Why:** It's the go-to Python library for classic machine learning and integrates seamlessly with Pandas and Streamlit.
  + **Implementation:** Train your classification model (e.g., RandomForestClassifier) in a Jupyter Notebook for experimentation, then save the final, trained model to a file (using joblib). Your Streamlit app will then load this pre-trained model file to make live predictions.
* **Showcasing LLM Skills (The "Wow" Feature): LangChain + Ollama**
  + **Why:** This adds a creative, generative AI component that makes your project truly unique.
  + **Implementation:** Add a third page to your Streamlit app called **"AI Project Generator."**
    1. The user selects a few technology categories from a dropdown (e.g., 'Frontend Framework', 'Cloud Platform').
    2. These selections are fed into a prompt template using **LangChain**.
    3. LangChain sends the prompt to your local **Ollama** model.
    4. The model generates a unique project idea, like: *"Idea: Build a serverless web app using React (Frontend Framework) that is deployed on AWS Lambda (Cloud Platform) to track real-time public transit data."*

**Phase 3: Infrastructure & Deployment**

**Goal:** Package your entire project professionally and deploy it online. This is a crucial step that many people skip.

* **Containerization: Docker + docker-compose**
  + **Why:** This is arguably one of the most important infrastructure skills for any data or AI engineer. It ensures your project can run anywhere, solves the "it works on my machine" problem, and is a massive plus for employers. It's easier to learn than you might think.
  + **Implementation:**
    1. Dockerfile: Create one for your Streamlit application. This file lists the instructions to build a portable image of your app.
    2. docker-compose.yml: This simple file can define and run your entire multi-container application: your Streamlit app and your Postgres database (if you choose to run it locally).
* **Deployment: Streamlit Community Cloud**
  + **Why:** It's the easiest, fastest, and free-est way to deploy a public-facing Streamlit app. It connects directly to your GitHub repository.
  + **Implementation:** Simply push your code to a public GitHub repo, connect your account to Streamlit Community Cloud, and deploy. Your app will be live in minutes.

**Summary: Your Recommended Tech Stack**

| Component | Technology | Why it's a great choice |
| --- | --- | --- |
| **Database** | **PostgreSQL on Supabase** | You know Postgres; Supabase adds a modern, easy-to-use cloud layer. |
| **Data Pipeline (ETL)** | **Python (Pandas, SQLAlchemy)** | Industry standard, leverages your core skills. |
| **AI-Powered Cleaning** | **Ollama + LangChain** | Advanced, impressive technique to showcase your AI skills in engineering. |
| **ML Model** | **Python (scikit-learn)** | The right tool for the job; standard and powerful. |
| **App/Dashboard** | **Streamlit** | You know it; perfect for building and deploying data apps in Python. |
| **Containerization** | **Docker** | A critical, highly-sought-after skill that shows professionalism. |
| **Deployment** | **GitHub + Streamlit Community Cloud** | The fastest path to getting a live, shareable portfolio link. |