Phase 1: Planning and Research

1. **Define Requirements**:
    - Input: CSV files containing PokerNow game logs.
    - Processing: Parse the logs to extract game details and player actions.
    - Output: Store data in a database and make aggregated data accessible to users.
    - Access Control: Users must upload 50+ hands to access data.
2. **Learn Basics**:
    - **Frontend**: HTML, CSS, and JavaScript.
    - **Backend**: Python (e.g., Flask/Django) or JavaScript (Node.js).
    - **Database**: SQL (SQLite for simplicity or MySQL/PostgreSQL for scalability).
3. **Research Hosting Options**:
    - Low-cost platforms like **Heroku**, **Vercel**, or **Replit** can host small projects for free.
    - Consider **Render** for free hosting with basic database integration.
4. **Data Security and Privacy**:
    - Plan how you'll secure user data (e.g., encryption, access controls).

Phase 2: Build MVP (Minimum Viable Product)

1. **Set Up a Basic Website**:
    - Create a simple interface using HTML/CSS.
    - Include a file upload form for CSV files.
2. **Parse CSV Files**:
    - Write a script to process uploaded CSVs.
    - Extract relevant data (e.g., player actions, hand outcomes).
    - Use Python's `csv` module or JavaScript libraries like `PapaParse`.
3. **Set Up a Database**:
    - Start with SQLite for ease of use.
    - Design tables to store game details, player actions, and user data (e.g., `users`, `hands`, `actions`).
4. **Implement User Accounts**:
    - Allow users to register and log in.
    - Use simple authentication libraries like Flask-Login (Python) or Passport.js (Node.js).
5. **Upload and Process CSVs**:
    - Build a backend route to handle file uploads.
    - Store parsed data in the database.
    - Track how many hands each user has uploaded.

Phase 3: Implement Data Access Features

1. **Access Control**:
    - Check if a user has uploaded at least 50 hands.
    - Grant access to data only if this condition is met.
2. **Data Display**:
    - Create a simple dashboard to visualize player tendencies.
    - Use charts and tables (libraries like Chart.js or D3.js for frontend visuals).
3. **Aggregate Data**:
    - Allow users to view trends, such as:
        - Most common actions by specific players.
        - Win rates by situation.
        - Their own tendencies.

Phase 4: Deployment and Scaling

1. **Deploy the Website**:
    - Use a free or low-cost platform for hosting.
    - Consider free-tier database options like ElephantSQL or Heroku Postgres.
2. **Monitor and Maintain**:
    - Set up logging and error tracking.
    - Test the site regularly to ensure stability.

Phase 5: Future Enhancements

1. **Improve UI/UX**:
    - Refine the website's appearance and usability.
    - Add responsive design for mobile users.
2. **Optimize Performance**:
    - Switch to a more robust database if the project scales (e.g., MySQL/PostgreSQL).
    - Implement caching for frequently accessed data.
3. **Add Advanced Features**:
    - Include data export options.
    - Allow users to filter and query the database.

Feasibility for Beginners

Challenges:

- **Learning Curve**: You'll need to learn the basics of web development, database management, and CSV parsing.
- **Data Security**: Ensuring data privacy and proper access control requires care.
- **Time**: Developing and testing the project will take time, especially while learning.

Advantages:

- Free hosting platforms and beginner-friendly frameworks/tools make this project feasible.
- It's modular—each step builds on the last, so you can work incrementally.

Learning Resources

- **Frontend**: freeCodeCamp
- **Backend**: Flask Official Docs or Node.js Guide
- **SQL**: SQL Tutorial
- **CSV Parsing**: Python's `pandas` or JavaScript's `PapaParse`