Got it! You want to approach the project like a **Business Analyst (BA)**, focusing on **requirements gathering, user stories**, and **breakdown of tasks** with more detailed **business goals**, **features**, and **deliverables**. Here's how we can break it down:

**Project: QueryCraft – Natural Language Data Analyst (No LLM)**

**Goal:**

To create a lightweight tool that allows users to ask natural language questions about tabular data (CSV) and receive answers both in textual form and as visualizations (e.g., charts).

**1. Stakeholder Requirements (Business Analysis)**

**1.1. Primary Stakeholders:**

* **End Users (Data Analysts/Business Analysts)**: Need to interact with data in a more intuitive way (no need to write complex SQL queries).
* **Developers**: Need a robust system to handle natural language input and query execution.
* **Managers**: Want insights on team performance, productivity, or sales data.

**1.2. Requirements Gathering:**

* **User Stories**:
  + **As an Analyst**, I want to upload a CSV file, so I can start analyzing my data.
  + **As an Analyst**, I want to ask simple natural language questions like “What is the average salary?” and get a clear, concise answer.
  + **As an Analyst**, I want to see trends over time (e.g., “How did sales perform last quarter?”).
  + **As a Developer**, I want the system to parse questions and map them to relevant columns in the dataset automatically.
  + **As an Analyst**, I want visual charts to represent the data (e.g., bar chart for top categories).

**2. Business Features & Functional Breakdown**

**2.1. Feature Set:**

**Core Features:**

* **Data Ingestion**: Upload a CSV file and load the dataset into the system.
* **Natural Language Processing (NLP)**:
  + Detect intent in user questions (e.g., aggregation, filtering, trend analysis).
  + Extract relevant column names and metrics from the question.
* **Data Query Execution**:
  + Generate SQL-like queries or pandas operations based on the parsed question.
  + Handle different types of questions (aggregation, sorting, filtering).
* **Data Visualization**: Automatically generate charts (e.g., bar charts, line charts) to represent query results.
* **Results Display**: Display both textual answers and corresponding charts on the UI.

**Advanced Features (Optional/Future Enhancements):**

* **Multi-Turn Conversations**: Remember previous queries to handle follow-up questions (e.g., "Show me trends for last quarter" → "Now break it down by region").
* **Data Export**: Allow users to download the results (in CSV/PDF).
* **Custom Query Support**: Allow users to input custom queries via a user-friendly interface.

**3. System Requirements & Constraints**

**3.1. Technical Requirements:**

* **Data Handling**: Use pandas for data loading, query execution, and basic aggregations.
* **Natural Language Processing**: Simple rule-based approach using libraries like spaCy or nltk to preprocess questions and detect intent.
* **Charting**: Use matplotlib or plotly for generating charts based on query results.
* **UI**: Streamlit for an easy-to-deploy, interactive user interface.

**3.2. Non-Functional Requirements:**

* **Performance**: System should handle typical-sized datasets (up to 100,000 rows) without significant lag.
* **Usability**: Must be easy for analysts to interact with — minimal setup required.
* **Scalability**: The system should be easily extensible to handle additional complex queries (multi-turn conversations, advanced visualizations).
* **Security**: Ensure that the data uploaded by users is stored securely (encryption of sensitive data).

**4. Use Case Scenarios (User Stories Breakdown)**

**4.1. Use Case 1: Data Upload**

* **User Action**: The user uploads a CSV file through the Streamlit UI.
* **System Response**:
  + Parse the CSV file into a pandas DataFrame.
  + Show a preview of the first few rows and display basic summary statistics (e.g., column names, data types, null values).
  + Allow the user to select which dataset they want to query.

**Acceptance Criteria:**

* The user can upload any CSV file with a maximum size of 10MB.
* The system parses the file correctly and displays a preview.
* The user can see the data types and a summary of the dataset.

**4.2. Use Case 2: Ask a Question (Aggregation)**

* **User Action**: The user asks a question like “What is the average salary?”.
* **System Response**:
  + Detect the intent as “aggregation”.
  + Identify the target column (salary).
  + Perform the aggregation using pandas (df['salary'].mean()).
  + Show the result as a text response and a simple bar chart (if applicable).

**Acceptance Criteria:**

* The system correctly identifies the aggregation intent and executes the query.
* The result is shown in a readable text format (e.g., “The average salary is $X”).
* A corresponding chart is shown if appropriate (e.g., bar chart for comparison).

**4.3. Use Case 3: Ask a Question (Trend)**

* **User Action**: The user asks a question like “Show me sales trends for 2023”.
* **System Response**:
  + Detect the intent as “trend”.
  + Identify the relevant columns (e.g., sales, date).
  + Group data by time (e.g., by month) and plot the trend using matplotlib or plotly.

**Acceptance Criteria:**

* The system extracts the correct time-related columns and visualizes trends over time.
* The chart should be interactive (if using plotly).

**4.4. Use Case 4: Ask a Question (Top-K)**

* **User Action**: The user asks “Show me the top 5 products by sales”.
* **System Response**:
  + Detect the intent as “top-k”.
  + Identify the product and sales columns.
  + Sort data by sales in descending order and return the top 5 products.
  + Display the result as a text answer and a bar chart.

**Acceptance Criteria:**

* The system correctly identifies the top 5 products and displays both the table and the chart.

**5. Project Milestones (Timeline & Deliverables)**

**5.1. Milestone 1: Project Setup and Data Ingestion**

* **Tasks**:
  + Create the project structure.
  + Set up data upload and preview functionality.
  + Extract basic summary statistics.
* **Deliverable**: Functional upload system with file preview.

**5.2. Milestone 2: NLP Question Parsing (Basic)**

* **Tasks**:
  + Implement basic text preprocessing (e.g., tokenization, lemmatization).
  + Develop the intent detection and column mapping logic.
* **Deliverable**: A working NLP pipeline that detects basic question intents and maps to columns.

**5.3. Milestone 3: Data Query Execution**

* **Tasks**:
  + Implement aggregation and filtering logic.
  + Set up basic query execution via pandas.
* **Deliverable**: Query execution system that processes user questions and returns data.

**5.4. Milestone 4: Visualization Integration**

* **Tasks**:
  + Integrate charts (bar, line, pie) with pandas query results.
  + Display text and charts together.
* **Deliverable**: Full functionality for generating text and visual answers.

**5.5. Milestone 5: User Interface (Streamlit)**

* **Tasks**:
  + Build a simple web interface for data upload and querying.
  + Display the results (text + charts).
* **Deliverable**: Interactive Streamlit app for querying CSV data.

**5.6. Milestone 6: Testing and Finalization**

* **Tasks**:
  + Test the app with real-world sample data.
  + Write unit tests and validate edge cases.
* **Deliverable**: Fully tested and stable application.

**6. Dependencies**

* **External Libraries**: pandas, nltk, spaCy, matplotlib, plotly, streamlit
* **Tools**: GitHub for version control, Heroku/Streamlit Sharing for deployment

**7. Risk Analysis**

* **Risk**: Parsing complex questions without an LLM could lead to inaccurate results.
  + **Mitigation**: Start with simple intent detection and improve as the system evolves.
* **Risk**: Handling large datasets may cause performance issues.
  + **Mitigation**: Implement optimizations like pagination or limiting data size.

With these **business goals, user stories, milestones, and acceptance criteria**, you’ll have a solid foundation for moving forward. Would you like to dive into any specific section or feature in more detail?