# Data Ingestion Pipeline for CockroachDB using FastAPI

## 1. Project Overview

This project is designed to facilitate seamless data ingestion from Excel files into CockroachDB using FastAPI. The solution dynamically handles database configurations, table creation, and batch-wise data insertion while ensuring scalability and reliability.

## 2. Pipeline Architecture

1. \*\*Input\*\*:

* - Users upload an Excel file via the FastAPI endpoint.  
  - Configurations such as database host, username, table name, and batch size are provided dynamically as part of the API request.

2. \*\*Processing\*\*:

* - The Excel file is parsed using Python’s `pandas` library into a structured DataFrame.  
  - The schema is inferred, and if the specified table does not exist in CockroachDB, it is created automatically.  
  - Data is inserted in batches to optimize performance.

3. \*\*Storage\*\*:

* - CockroachDB, a distributed SQL database, stores the ingested data.

4. \*\*Output\*\*:

* - A JSON response indicating the number of rows successfully inserted into the database.

## 3. Services and Technologies Used

1. \*\*FastAPI\*\*:

- Lightweight web framework to expose RESTful APIs for data ingestion.  
- Provides excellent performance and support for asynchronous operations.

2. \*\*Pandas\*\*:

- Handles Excel file parsing and preprocessing.

3. \*\*CockroachDB\*\*:

- A scalable, distributed SQL database optimized for high availability and low latency.  
- CockroachDB Cloud Free Cluster is used for hosting the database.

4. \*\*SQLAlchemy\*\*:

- Simplifies database interactions and query execution.

5. \*\*Uvicorn\*\*:

- ASGI server for running FastAPI applications.

6. \*\*Deployment Tools\*\* (Optional):

- \*\*Docker\*\*: For containerizing the application for consistency and portability.  
- \*\*Cloud Hosting\*\*: AWS/GCP/Azure or other hosting services for deploying the application.

## 4. Suggested Pipeline Workflow

Step 1: User uploads an Excel file with configuration parameters.  
Step 2: API validates the file format and configuration details.  
Step 3: File data is processed into a DataFrame using pandas.  
Step 4: Table schema is validated or created dynamically in CockroachDB.  
Step 5: Data is inserted in batches, ensuring efficient use of resources.  
Step 6: A response with the count of inserted rows is sent back to the user.

## 5. Suggestions for Optimization

1. \*\*Data Validation\*\*:

- Add schema validation to check if the uploaded Excel file matches the expected format.  
- Use libraries like `pydantic` for parameter validation.

2. \*\*Security\*\*:

- Use HTTPS for secure API communication.  
- Store database credentials securely (e.g., environment variables or a secrets manager).

3. \*\*Scalability\*\*:

- Incorporate a queuing system like RabbitMQ or Kafka for high-volume data uploads.  
- Use CockroachDB's built-in horizontal scaling features for handling large datasets.

4. \*\*Monitoring and Alerts\*\*:

- Implement monitoring tools like Prometheus or Grafana for real-time insights into API and database performance.  
- Configure alerts for failures or high latency.

5. \*\*Error Handling\*\*:

- Ensure comprehensive error handling for database connectivity, file parsing, and schema mismatches.  
- Provide meaningful error messages to the API user.

6. \*\*Deployment\*\*:

- Use containerization (e.g., Docker) to deploy the application in a consistent environment.  
- Host the app close to the CockroachDB cluster to minimize latency.

7. \*\*Documentation\*\*:

- Provide API documentation using FastAPI’s built-in OpenAPI documentation.  
- Write clear instructions for clients on how to use the endpoints.

## 6. Visual Representation

Include a pipeline diagram to visualize the workflow:  
- User → FastAPI → Pandas (Data Processing) → CockroachDB (Storage) → JSON Response.