
SCHEMASPHERE
A SQL PLAYGROUND FOR YOUR CSV FILES!

🚀 KEY FEATURES & HIGHLIGHTS

01

Thoughtful & Efficient Design

✅ Well-planned features for real-world use.
✅ Supports SQL queries with fast execution.
✅ Loads & processes large CSV datasets smoothly.

02

User-Centric & Simple

✅ No database setup needed – works instantly.
✅ Clean, intuitive interface with quick navigation.
✅ Provides clear query results & execution time tracking.

03

Smart Performance Optimization

✅ Uses in-memory SQLite for speed.
✅ Handles compressed files efficiently.
✅ Fast response time even for large datasets.

04

Intelligent Error Handling

✅ Real-time logging for debugging.
✅ Tracks execution time to optimize queries.
✅ User-friendly error messages for smooth experience.

05

Clean, Readable & Scalable Code

✅ Well-structured, modular, and easy to maintain.
✅ Follows Python best practices for efficiency.
✅ Lightweight & scalable for future improvements.

06

Snappy, Fast & Responsive

✅ No lag – queries run instantly.
✅ Optimized data fetching & processing.
✅ Works seamlessly across different datasets.

🛠️ TECHNOLOGIES & TOOLS USED

🎨 Frontend (User Interface & Experience)

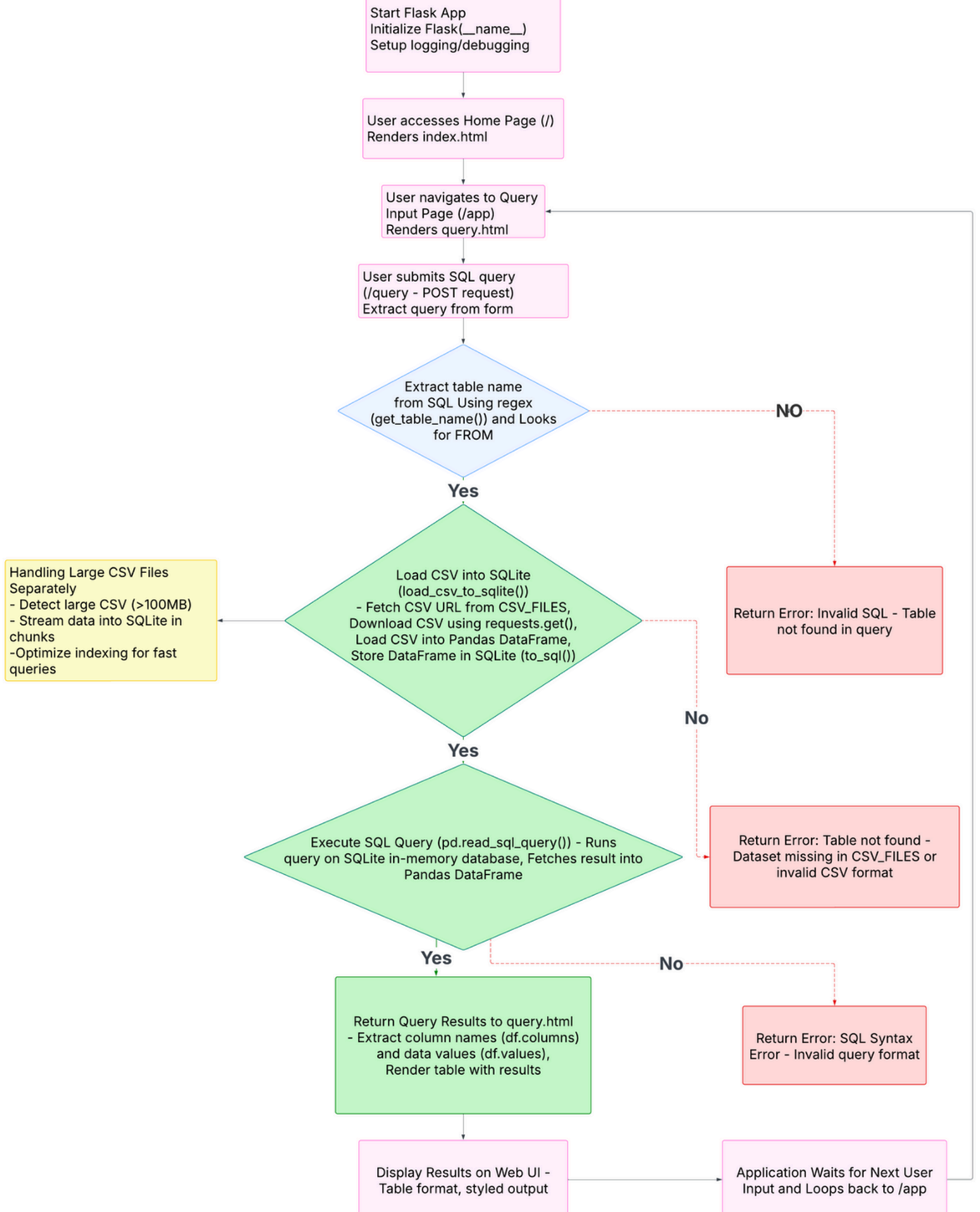
- HTML5 – Structure of the web interface.
- CSS3 – Styling for the UI.

🔥 Backend (Server-Side Processing & Query Execution)

- Flask – Python web framework handling HTTP requests and SQL query execution.
- SQLite (in-memory) – Temporary database for fast, lightweight query execution.
- Pandas – Used for reading, processing, and querying CSV data.
- Requests – Fetches remote CSV files from URLs.
- gzip & zlib – Automatic decompression support for gzip-compressed CSV files.

🚀 Deployment & Hosting

- Render – Cloud hosting platform used for deploying SCHEMASPHERE.
- Gunicorn – WSGI server for running the Flask app in production.
- GitHub – Code repository and version control.



WORKING

1 Start Flask App → Initialize & debug

2 Home Page (/) → Load index.html

3 Query Page (/app) → Load query.html

4 User Uploads Large File (/upload) → Save to server

5 Validate File → Check type & size

6 Chunked Processing → Read & process in parts

7 Store in SQLite → Load CSV (batch inserts)

8 Submit SQL (/query) → Extract & validate query

9 Extract Table Name → If missing, return error

10 Run SQL Query → Process stored data (handle syntax errors)

11 Return Results → Display as a styled table

12 Loop Back → Wait for next query/file

CHALLENGES ENCOUNTERED

01

Handling Large Files

- Large CSV uploads caused memory spikes, leading to crashes.
- Reading entire files at once slowed processing.
- Implemented chunked reading & batch inserts into SQLite.

02

Error Handling

- SQL queries missing FROM <table_name> caused extraction failures.
- CSV files with missing headers or incorrect formats led to parsing errors.
- Added validation checks for queries and file formats.

03

Concurrency Issues

- Multiple users querying simultaneously caused SQLite database locks.
- Single-threaded execution delayed responses.
- Optimized queries and used connection pooling for better handling.

📊 SAMPLE DATASET OVERVIEW & SQL QUERIES FOR SCHEMASPHERE

📌 Datasets Used:

- Northwind Database (Relational dataset with orders, customers, employees, products, etc.)
- Large CSV Dataset (More than 100,000 rows with fields like Customer ID, Name, Company, City, Country, Phone, Email, Subscription Date, etc.)

🔍 Sample Queries

1️⃣ SELECT * FROM "large" ;
👉 Fetches the first five records.

Row count : 1,00,000

Query Results:

88	C8CD21E646F878f	Kaylee	Goodman	Joseph, Stevens and Webster
89	52DCA69f8E6CEE4	Sally	Blackwell	Hart, Sanfor and Fernandez
90	B62eCbba2aF49eE	Beverly	Gamble	Pitts-Bautista
91	7f1Cd9CCdbAeCFf	Denise	Pollard	Oneal and Sons
92	D4f427290f8dE7F	Ryan	Solis	Cooper Inc
93	eC8d7bDF085FcD5	Isaac	Bradshaw	Mcintosh-

Execution Time: 0.316614 seconds seconds