



INTEGRATING SUPERSET USING NPCYF

A PROJECT REPORT submitted for Data Science Internship at ISI,KOLKATA



By

Avimalya Dey



OCTOBER 31, 2024

IDEAS - INSTITUTE OF DATA ENGINEERING, ANALYTICS AND SCIENCE FOUNDATION
TECHNOLOGYINNOVATION HUB @ INDIAN STATISTICAL INSTITUTE, KOLKATA203 BARRACKPORE
TRUNK ROAD, KOLKATA-700108, INDIA

Project Overview

The objective of this project is to create a comprehensive dashboard to visualize and analyse crop production data in India. The dashboard provides insights into various crop production metrics over time, enabling stakeholders to make informed decisions. Building interactive dashboards using Apache Superset by connecting to SQL databases and embedding them into a front-end platform. The project covers installation, data connection, dashboard creation, and integration, with optional exploration of advanced features like time series and predictive analysis.

Dataset overview

The crop production dataset provides annual estimates (in lakh tones) for various crops across different seasonal categories—Kharif, Rabi, and Summer—from 2013 to 2024. Compiled by the Ministry of Agriculture & Farmers Welfare (DA&FW), the dataset includes columns for crop types, seasonal categories, and yearly production data, primarily in numerical format. Cleaning steps include addressing missing data (indicated by symbols like "@", "\$", "#") through imputation or exclusion, ensuring unit consistency, and checking for anomalies or outliers. This ensures data readiness for accurate analysis and visualization.

This project demonstrates the successful integration of Apache Superset for the visualization and analysis of crop production data, underpinned by a MySQL backend and secured through robust role-based user access controls. By embedding interactive dashboards, it delivers accessible, data-driven insights into crop production trends and price correlations, empowering users with an intuitive exploration of agricultural metrics. Future developments could explore expanding data sources, advancing analytical capabilities, and enhancing the user experience through sophisticated filtering options and additional customization features. These enhancements would deepen insights and further support data-informed decision-making in crop management and agricultural policy.

Preprocessing of Data

In the final dataset, multiple data cleaning steps were performed to enhance clarity, accuracy, and consistency. Missing data, indicated by symbols like "@", "\$", and "*", was addressed through imputation or by excluding irrelevant entries. Records for crops such as 'Jute ##', 'Mesta ##', 'Total Pulses', 'Cereals', 'Total Food Grains', and 'Total Oil Seeds' were removed to eliminate redundancy. Production values for 'Jute & Mesta ##*' and 'Cotton#' were converted from lakh bales to tones using appropriate multipliers, and rounded to two decimal places for uniformity. Crop names were updated for clarity (e.g., 'Jute & Mesta ##*' to 'Jute and Mesta'). Missing production values for certain crops in specific seasons were replaced with 0 to maintain data consistency across all years.

Table 1: Sample Dataset

Crop	Season	Year	Production
Rice	Kharif	2013	914.97
Rice	Rabi	2013	151.49
Rice	Summer	2013	0
Rice	Kharif	2014	913.92

Installation of Apache Superset, SQL Workbench using Docker

Installation of Docker

1. **Download Docker Desktop:** Downloaded docker desktop from [Docker Official Site](https://docs.docker.com/docker-for-windows/install/)
2. **Run the Installer:** Executed the downloaded installer file as an administrator.
3. **Configuration:** enabled WSL 2, for optimal performance.
4. **Restarted:** After the installation was complete, restarted the application.
5. **Launch Docker:** Open Docker Desktop to finish setup.
6. **Verification:** Verified if docker was installed successfully by running `docker --version` in powershell.

Installation of SQL Workbench & Apache Superset

In setting up MySQL, Docker Desktop, Git, and Apache Superset, I began by installing MySQL Server and Workbench, creating a database, setting up tables, inserting sample records, and performing essential queries. I also imported data from a CSV file into the database.

Next, I installed Docker Desktop and Git. After pulling the Superset image from Docker, I ran it in a container. I confirmed the installation by checking Docker's version and verifying the image tag. I launched Superset with:

```
docker run -d -p 8080:8088 -e "SUPERSET_SECRET_KEY=mysuperset" --name superset  
apache/superset:27ff8f2-dev
```

Then, I created an admin user with the following command:

```
docker exec -it superset superset fab create-admin --username admin --firstname  
Superset --lastname Admin --email admin@superset.com --password admin
```

Finally, I upgraded the Superset database and loaded example datasets with:

```
docker exec -it superset superset db upgrade  
docker exec -it superset superset load_examples
```

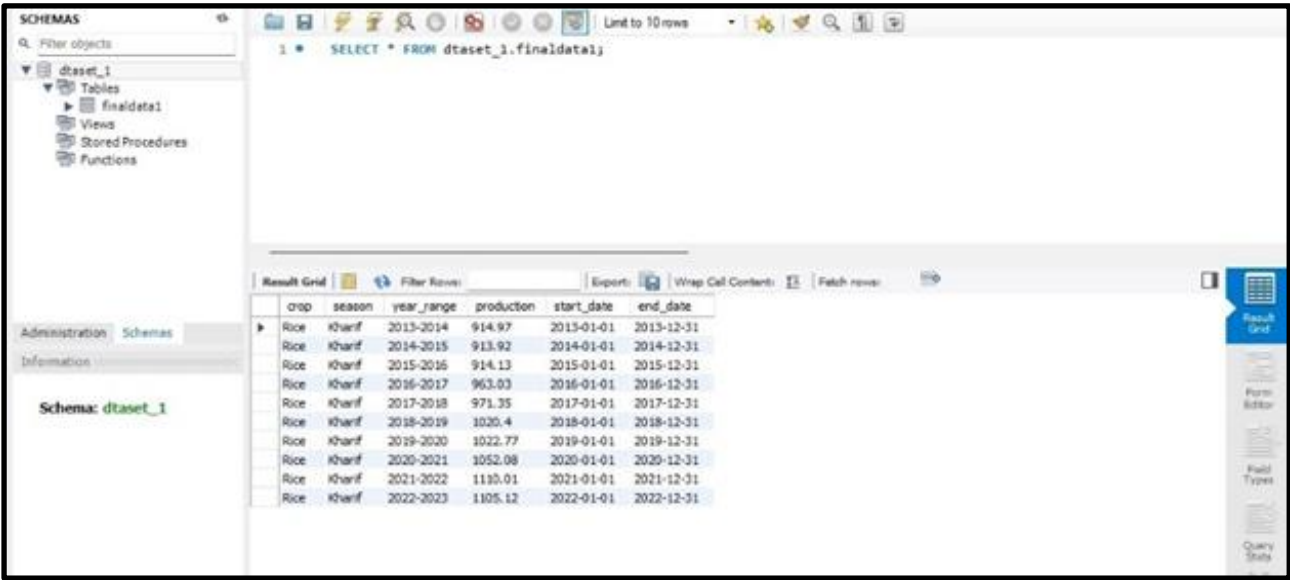
This completed the setup, allowing me to access Superset for analytics tasks

Dashboard Creation and design

Data import and setup

The data preprocessing phase began in excel, where raw data was meticulously cleaned and structured for compatibility, ultimately saved as a .csv file for further use. This clean dataset was then imported into SQL Workbench, where a dedicated connection named "finaldata1" was created for this project. Within this connection, a specific schema labeled "dtaset_1" was set up to organize project data efficiently. Following this import, a database connection was established within Apache Superset, seamlessly linking it to the SQL Workbench instance to facilitate the data's integration into Superset. Within Superset, further data transformation steps were applied directly to the `finaldata1` dataset. Two calculated columns—`Crop` and `season`—were created to enrich the dataset and optimize it for analysis. To enable time-series analysis and facilitate the construction of line charts, the `Year` column, initially stored as an integer, was converted to datetime format.

Table 2: SQL Database



The screenshot shows the Apache Superset interface. On the left, the 'SCHEMAS' panel displays a tree view with 'dtaset_1' expanded, showing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'Tables' section is further expanded, showing 'finaldata1'. The main panel displays a SQL query: 'SELECT * FROM dtaset_1.finaldata1;'. Below the query, the 'Result Grid' shows a table with 10 rows and 6 columns: 'crop', 'season', 'year_range', 'production', 'start_date', and 'end_date'. The data is as follows:

crop	season	year_range	production	start_date	end_date
Rice	Kharif	2013-2014	914.97	2013-01-01	2013-12-31
Rice	Kharif	2014-2015	913.92	2014-01-01	2014-12-31
Rice	Kharif	2015-2016	914.13	2015-01-01	2015-12-31
Rice	Kharif	2016-2017	963.03	2016-01-01	2016-12-31
Rice	Kharif	2017-2018	971.35	2017-01-01	2017-12-31
Rice	Kharif	2018-2019	1020.4	2018-01-01	2018-12-31
Rice	Kharif	2019-2020	1022.77	2019-01-01	2019-12-31
Rice	Kharif	2020-2021	1052.08	2020-01-01	2020-12-31
Rice	Kharif	2021-2022	1130.01	2021-01-01	2021-12-31
Rice	Kharif	2022-2023	1105.12	2022-01-01	2022-12-31

Dashboard Design

The dashboard consists of two main tabs: Overall Crop Production view and specific production view .

Overall Crop Production and Specific production Analysis

The crop production dashboard presents a comprehensive view of agricultural trends from 2013-2014 to 2023-2024. A bar chart reveals a steady increase in total production, peaking around 2022-2023, while a stacked bar chart highlights seasonal patterns, with Kharif dominating at 5-7k units and Rabi steady at around 2k units, while summer production remains minimal. A pie chart shows crop distribution, with sugarcane leading, followed by rice, wheat, and nutri/coarse cereals. The line chart illustrates a decade-long upward trend in total crop production, from 6k to 9k units. Finally, a heatmap visualizes production intensity for various crops, with

crops like urad, sunflower, and soybean showing strong seasonal output. Together, these visualizations capture key insights into production trends, seasonal variations, and crop-specific performance.

The specific view dashboard for crop production analysis highlights seasonal variations and crop performance metrics over several years. The left graph in Image 1 shows a bar chart of total production (SUM) from 2013-2014 to 2023-2024, indicating a clear upward trend, particularly in recent years with values peaking around 8,000 to 9,000. The right graph presents a multi-series bar chart comparing the highest and lowest production years for various crops, with Kharif season production (light blue bars) generally exceeding Rabi (dark blue) and another season (green). In Image 2, a heat map illustrates crop performance across seasons, while a table summarizes production metrics, showing values from approximately 3,000 to 4,400 units and percentage etrics ranging from 7% to 10%. Overall, the dashboard provides a concise view of crop production patterns and performance over time.

Filters and Interactivity

The dashboard features three interactive filters that enhance data analysis. The Season Filter allows users to select between Rabi, Kharif, or Summer seasons for targeted insights, while the Year Filter enables filtering of data by specific years from 2013-14 to 2022-23, highlighting trends over time. Additionally, the Crop Type Filter allows users to focus on individual crop production trends.

FIGURE 1 : Filters in Apache

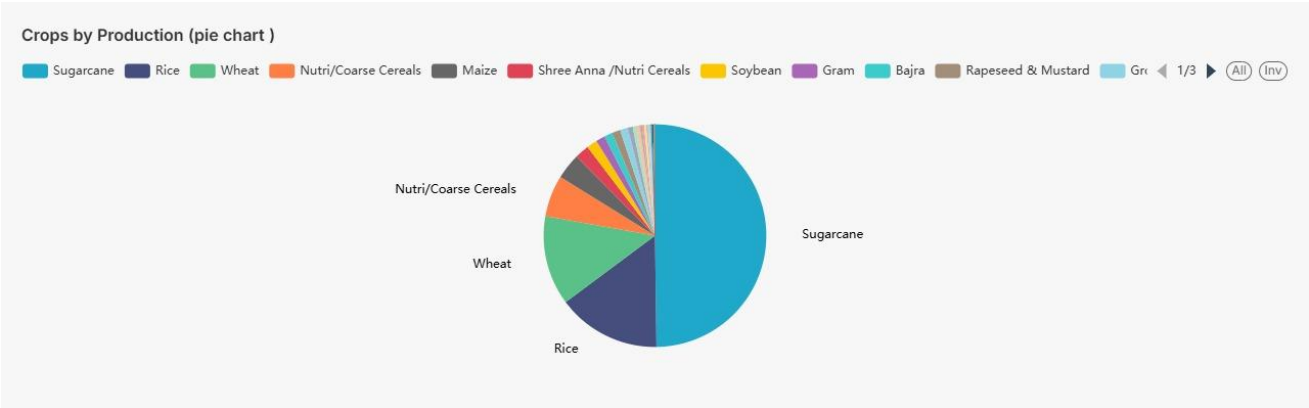
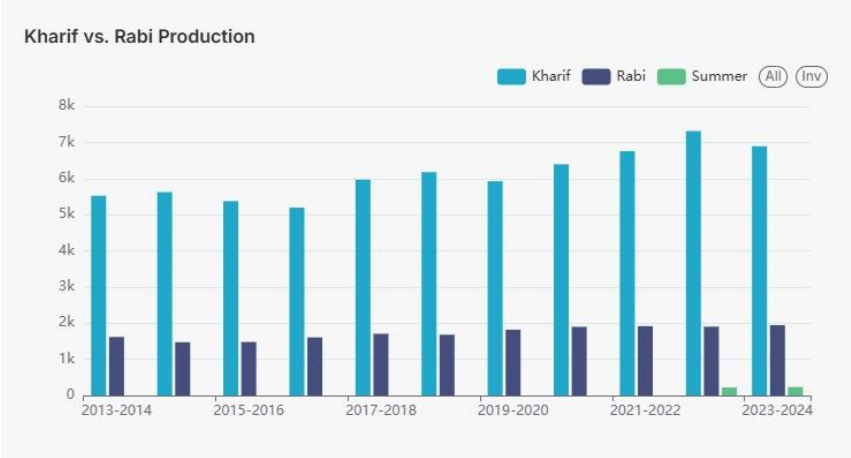
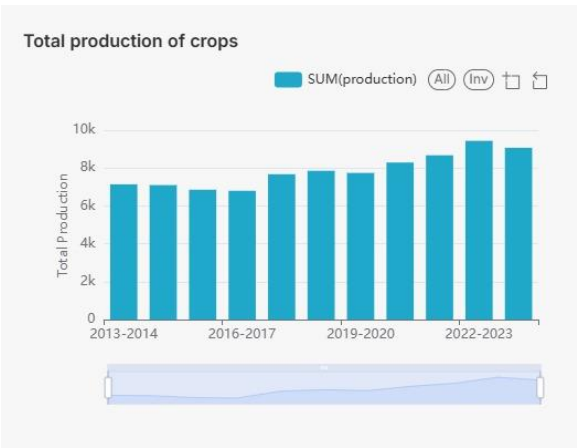
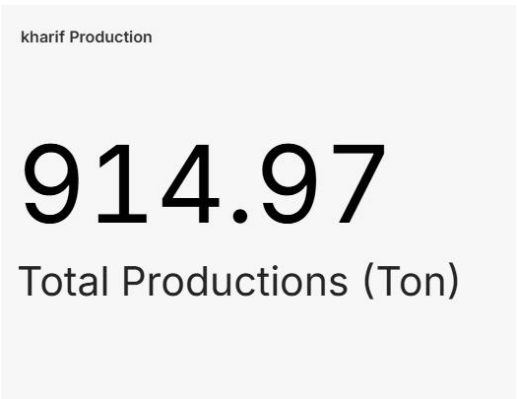


Table 3: Visualization Charts

Visualization name	Chart type	Dashboard	Dataset used
Total production of crops	Barchart	Overall view	finaldata_1
Kharif vs Rabi Production	Histogram	Overall view	finaldata_1
Crops production (%)	Pie chart	Overall view	finaldata_1
Crops breakdown over time	Area chart	Overall view	finaldata_1
Production by crop and season	Heatmap	Overall view	finaldata_1
Kharif production	Big number	Overall view	finaldata_1
Seasonal Breakdown	Bar chart	Overall view	finaldata_1
Highest and LOWEST PRODUCTION YEARS FOR EACH	Histogram	SPECIFIC VIEW	finaldata_1
SEASON WISE BEST PERFORMING CROPS	Heatmap	SPECIFIC VIEW	finaldata_1
Percentage Chart	Table	SPECIFIC VIEW	finaldata_1

List of Visualizations

A comprehensive table of all charts used across the dashboards, including their visualization names, chart types, associated dashboards, and datasets, is provided in the next on the following page. Moreover, screenshots of the dashboard are also attached.



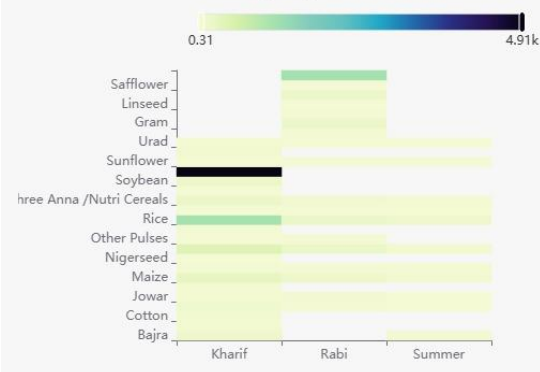
Crops Breakdown Over Time



Heatmap (Production by Crop and Year)



Season-wise Best Performing Crop



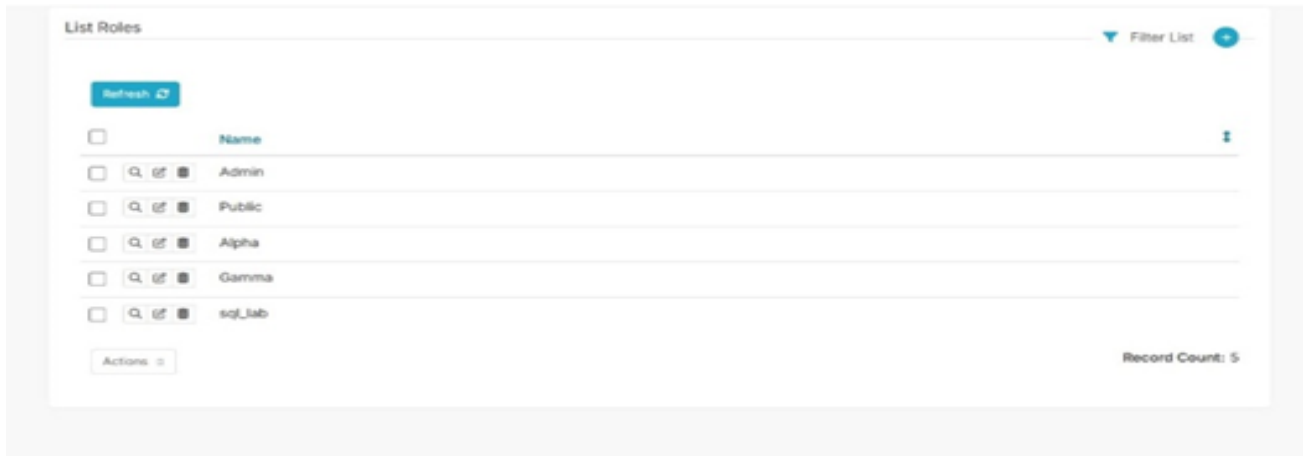
percentage chart

production	SUM(crop)	%MAX(production)	%MIN(production)
3521.42	0	8.164%	8.164%
3623.33	0	8.400%	8.400%
3484.48	0	8.078%	8.078%
3060.69	0	7.096%	7.096%
3799.05	0	8.808%	8.808%
4054.16	0	9.399%	9.399%
3705	0	8.590%	8.590%
4053.99	0	9.399%	9.399%
4394.25	0	10.188%	10.188%
Summary	0		

Role based Access used

In our project, we created a custom role, Public, which extends the permissions of the Gamma role and includes additional access specifically tailored to the "Crop Production" dashboard. Overall, five roles were configured within the project: Admin, Gamma, Alpha, SQL Lab, and Public. To manage access, we established two distinct user profiles in the system. The first is the default Admin user, with comprehensive permissions across the platform. The second is a secondary user, assigned both the Public and Gamma roles, providing them with controlled, limited access to the crop production dashboard. Below are screenshots illustrating the role configurations and user assignments, supporting our permissions framework.

Table 4: Role on superset



<input type="checkbox"/>	Name	
<input type="checkbox"/>	Admin	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	Public	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	Alpha	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	Gamma	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/>	sqs_lab	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

Dashboard Embedding using i-frame

To embed the dashboard, we utilized the HTML ``<iframe>`` tag, allowing seamless integration within the project's web page. Prior to embedding, we configured the necessary settings in a `superset_config.py` file, which was created and placed in the `superset/docker/pythonpath_dev` directory. This configuration enabled the required permissions and settings to support dashboard embedding functionality.

```
FEATURE_FLAGS = {  
    "EMBEDDED_SUPERSET": True,  
}  
ENABLE_PROXY_FIX = True  
SESSION_COOKIE_SAMESITE = None  
PUBLIC_ROLE_LIKE_GAMMA = True  
AUTH_ROLE_PUBLIC = 'Gamma'  
WTF_CSRF_ENABLED = False  
TALISMAN_ENABLED = False
```

The `superset_config.py` file was meticulously configured to enable embedding of Superset dashboards within external applications by adjusting various security settings and access controls. Key configurations are outlined below:

FEATURE_FLAGS: This dictionary contains toggles for specific Superset features. Enabling `"EMBEDDED_SUPERSET": True` allows Superset dashboards or views to be embedded within external web pages or applications.

ENABLE_PROXY_FIX: Set to `True`, this setting enables Superset to handle headers correctly when deployed behind a proxy or load balancer, ensuring accurate recognition of headers, such as the original IP address.

SESSION_COOKIE_SAMESITE: Setting this to `None` removes the SameSite attribute from session cookies, facilitating cross-site requests essential for embedding Superset in other applications.

PUBLIC_ROLE_LIKE_GAMMA With this set to `True`, anonymous (public) users are granted the same permissions as those in the Gamma role, typically allowing basic dashboard viewing. This setting simplifies embedding by removing the need for user authentication.

WTF_CSRF_ENABLED: Setting this to `False` disables Cross-Site Request Forgery (CSRF) protection, making embedding easier at the cost of certain security protections.

TALISMAN_ENABLED: Disabling Talisman by setting it to `False` reduces enforced HTTP security headers, such as those for clickjacking protection, allowing for smoother embedding but with reduced security measures.

HTTP_HEADERS Setting `X-Frame-Options` to `ALLOWALL` permits Superset to be embedded in iframes on any site. This configuration is critical for enabling embedding but should be used cautiously, as it can expose the application to potential clickjacking risks if not carefully managed.

Each setting in `superset_config.py` serves a specific role in facilitating the secure and effective embedding of Superset dashboards within external applications.

The following HTML code was used to embed the Apache Superset dashboard for the Project. It includes an iframe tag to display the dashboard and some basic styling for the page layout.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <p> Crop production dashboard </p>
  <iframe src="http://localhost:8080/superset/dashboard/p/PxIMk67LowR/" width="1600" height="1500"
  frameborder="0"></iframe>
</body>
</html>
```

Conclusion and Future Exploration

This project demonstrates the successful integration of Apache Superset for the visualization and analysis of crop production data, underpinned by a MySQL backend and secured through robust role-based user access controls. By embedding interactive dashboards, it delivers accessible, data-driven insights into crop production trends and price correlations, empowering users with an intuitive exploration of agricultural metrics. Future developments could explore expanding data sources, advancing analytical capabilities, and enhancing the user experience through sophisticated filtering options and additional customization features. These enhancements would deepen insights and further support data-informed decision-making in crop management and agricultural policy.

[Link to GitHub Repo](#)

-----END-----