# **Agriculture Data Analytics Project Documentation**

**Project Title:**

**State-wise and District-wise Agricultural Production & Yield Analysis**

## **Objective:**

To extract actionable insights from historical agricultural data using:

* Python for cleaning and exploratory analysis
* SQL for analytical querying
* Power BI for interactive data visualization

## **Dataset Overview:**

* **Dataset Name:** agri\_cleaned (final transformed dataset)
* **Format:** CSV ➔ PostgreSQL table
* **Columns:** 80 (including area, production, yield metrics)
* **Temporal Range:** Multi-year time series data
* **Spatial Coverage:** State and District level across India

## 

## **Data Processing and Cleaning (Python)**

### **Tasks Completed:**

* Converted units:  
  + Area from 1000 ha ➔ ha
  + Production from 1000 tons ➔ tons
  + Yield from kg/ha ➔ tons/ha
* Replaced negative values with NaN
* Verified post-cleaning columns count (retained 80)

**Key Python Snippets:**

for col in df\_clean.columns:

if 'AREA (1000 ha)' in col:

df\_clean[col.replace('AREA (1000 ha)', 'AREA (ha)')] = df\_clean[col] \* 1000

if 'PRODUCTION (1000 tons)' in col:

df\_clean[col.replace('PRODUCTION (1000 tons)', 'PRODUCTION (tons)')] = df\_clean[col] \* 1000

if 'YIELD (Kg per ha)' in col:

df\_clean[col.replace('YIELD (Kg per ha)', 'YIELD (tons per ha)')] = df\_clean[col] / 1000

# Drop original columns

df\_clean.drop(columns=[...], inplace=True)

### **Challenges:**

* String fields present in numeric columns
* NaNs introduced post conversion

### **Solutions:**

* Type coercion with pd.to\_numeric(errors='coerce')
* Revalidated dataset dimensions

**Exploratory Data Analysis (Plotly - Python)**

### **Key Insights Visualized:**

* Top rice and wheat-producing states
* Sunflower and oilseed production trends
* Sugarcane production over decades
* Area vs. Production scatter plots
* Rice vs. Wheat yield comparisons

### **Charts Used:**

* Line plots, Bar charts, Pie charts
* Scatter plots with regression lines
* Stacked Area Chart for time series

### **Sample Code:**

import plotly.graph\_objects as go

fig = go.Figure()

fig.add\_trace(go.Scatter(...))

fig.update\_layout(title='Rice vs Wheat Production')

fig.show()

## 

## 

## 

## **SQL Analysis (PostgreSQL)**

### **Sample Queries:**

1. Top 3 Rice Producing States by Year:

WITH ranked\_production AS (

SELECT "Year", "State Name", SUM("RICE PRODUCTION (tons)") AS total\_rice,

RANK() OVER (PARTITION BY "Year" ORDER BY SUM("RICE PRODUCTION (tons)") DESC) AS rank

FROM agri\_cleaned

GROUP BY "Year", "State Name"

)

SELECT \* FROM ranked\_production WHERE rank <= 3

1. Oilseed Production Growth (5-Year):

SELECT "State Name",

ROUND(((MAX("OILSEEDS PRODUCTION (tons)") - MIN("OILSEEDS PRODUCTION (tons)"))::numeric

/ NULLIF(MIN("OILSEEDS PRODUCTION (tons)")::numeric, 0)) \* 100, 2) AS growth

FROM agri\_cleaned

WHERE "Year" BETWEEN (SELECT MAX("Year") - 4 FROM agri\_cleaned) AND (SELECT MAX("Year") FROM agri\_cleaned)

GROUP BY "State Name"

ORDER BY growth DESC

LIMIT 5

### **Challenges:**

* PostgreSQL ROUND() casting issues
* Complex filtering over years and partitions

### **Fixes:**

* Explicit casting with ::numeric
* Simplified logic using CTEs

## **Power BI Visualization**

### **Steps:**

* Connected PostgreSQL with **Import mode**
* Each SQL query imported as a separate table
* Used visuals like:  
  + Bar Charts, Line Charts, Stacked Area Charts
  + Pie Charts, Clustered Columns

### **Visualizations:**

| **Query** | **Chart Type** |
| --- | --- |
| Top 3 Rice States | Stacked Area Chart |
| Oilseed Growth | Bar Chart |
| Yield Increase | Column Chart |
| Cotton Trend | Line Chart |
| District Rice Yield | Bar Chart |

### **Challenges:**

* Field name mapping errors
* Syntax errors in SQL editor
* Missing fields in visuals

### **Fixes:**

* Added Year and State as categorical fields
* Used visual level filters in chart panels

**Overall Insights**

* Rice and Wheat production are heavily concentrated in a few northern states
* Oilseed production has shown substantial growth in central India
* Yield trends have improved steadily, but inconsistently across districts
* District-level analysis helps identify underperforming regions

## **Future Scope**

* Add ML forecasting (Prophet / XGBoost)
* Streamlit app with real-time filters
* Geospatial charts using Power BI Map or Plotly Express

## **Tools Used**

* Python (pandas, plotly)
* PostgreSQL (for storage and query)
* Power BI (for visualization)