# Task 01

## Apache Spark

Apache Spark is an impressive open-source computing system that's specifically designed to tackle big data challenges. What sets it apart is its ability to perform in-memory computations, making it significantly faster than traditional disk-based systems such as Hadoop MapReduce. As a result, Spark has emerged as a favorite among professionals in areas like data engineering, machine learning, and real-time analytics.

**Key Features of Apache Spark**

- In-Memory Processing – This feature allows data to be stored in RAM, which speeds up computation significantly.
- Multi-Language Support –. Apache Spark is versatile and can work with several programming languages, including Python (PySpark), Scala, Java, and R.
- Cluster Computing – It efficiently distributes processing tasks across multiple nodes, making it scalable and powerful.
- Unified Data Processing – Spark supports a variety of data processing methods, including batch processing, streaming, SQL queries, machine learning (MLlib), and graph processing (GraphX).
- Integration with Cloud Platforms – You can run Spark seamlessly on popular cloud services like AWS, Azure, and Google Cloud.

| ADVANTAGES | DESCRIPTION |
|---|---|
| Speed | Processes data 100x faster than Hadoop MapReduce due to in-memory execution. |
| Scalability | You can run it on just one machine or scale it up to thousands of nodes in a cluster. |
| Ease of Use | Discover straightforward APIs designed for big data applications, whether you're working with Python, Scala, or Java. |
| Fault Tolerance | It utilizes Resilient Distributed Datasets (RDDs) for recovery. |
| Versatility | Discover the power of our tool that supports batch processing, real-time streaming, machine learning, and graph analytics. Just paste your text, and in seconds, you'll receive accurate, human-like results! |
| Cost Efficiency | Our tool helps cut down on computing costs by handling large datasets in a much more efficient way. |

**Use Cases of Apache Spark**

- Financial Services – Risk analysis.
- E-Commerce – Personalized customer recommendations and real-time analytics.
- Healthcare – It helps with predictive analytics to enhance patient care.
- IoT and Sensor Data – . Perfect for handling real-time data from all those connected devices.

## Snowflake

Snowflake is a cloud-based data warehousing platform that's built to manage large amounts of data for storage, processing, and analytics. What sets Snowflake apart from traditional data warehouses is its ability to separate compute and storage, which means businesses can scale their resources independently as needed.

**Key Features of Snowflake**

- Cloud-Native Architecture – It's designed to run seamlessly on AWS, Microsoft Azure, and Google Cloud.
- Automatic Scaling – This feature automatically adjusts resources to meet the demands of your workload.
- Data Sharing – It facilitates secure, real-time data sharing between different organizations.
- Zero Management – Snowflake takes care of infrastructure, tuning, and optimization all on its own.
- Time Travel & Cloning – You can easily restore data to earlier versions and create instant copies whenever you need them.

| Advantages | Description |
|---|---|
| Separation of Storage & Compute | This feature helps optimize both performance and costs by allowing you to scale storage and computing resources independently. |
| Multi-Cloud Support | Snowflake is designed to work seamlessly across AWS, Azure, and GCP, giving you the flexibility you need. |
| Secure & Governed Data Sharing | It allows for real-time collaboration without the hassle of moving data around. |
| High Performance | With auto-scaling, caching, and query optimization, Snowflake ensures top-notch performance. |
| Pay-as-You-Go Pricing | This model helps keep costs down by charging you only for the resources you actually use. |

**Use Cases of Snowflake**

- Business Intelligence – Dive into customer behavior and spot trends.
- ETL (Extract, Transform, Load) Pipelines – Seamlessly move and transform massive datasets.
- Cybersecurity & Fraud Detection – Keep an eye out for threats and unusual activities.
- IoT & Big Data Analytics – Efficiently store and analyze fast-paced IoT data.

| Apache Spark | Snowflake |
|---|---|
| A powerful framework for processing big data | A cloud-based data warehouse solution |
| Handles batch jobs, real-time streaming, and machine learning | Utilizes SQL for analytics |
| Works with structured, semi-structured, and unstructured data | Mainly focuses on structured data |
| Delivers speed through in-memory execution | Designed for analytics but relies on disk storage |
| Great for data transformation, real-time processing, and AI/ML tasks | Perfect for data storage, reporting, and querying needs |

# Introduction and Explanation of the Dataset

## Introduction

When it comes to big data analytics, being able to process and analyze real-world datasets is essential for gaining actionable insights and guiding decision-making. In this final report, we dive into the Karnataka agricultural dataset (data_season.csv), which is a rich collection of agricultural data gathered over several years and regions in Karnataka, India. This dataset highlights important agronomic and environmental factors related to crop production, focusing mainly on various crops like coconut, coffee, cocoa, cardamom, pepper, and more, all across different seasons and locations. As part of our coursework for PUSL3121 C2 – Big Data Analytics, this dataset lays the groundwork for our preprocessing, analysis, and modeling efforts, allowing us to explore patterns in crop yields, environmental impacts, and economic results. Given that agriculture is a vital part of Karnataka's economy—employing a large segment of the population and contributing to both local food supply and export markets—gaining insights from this dataset presents valuable opportunities to leverage data-driven methods to boost agricultural productivity and sustainability.

The dataset includes 3,151 records, each representing a distinct agricultural observation from various districts such as Mangalore, Kodagu, Kasaragodu, Raichur, Gulbarga, Hassan, Madikeri, Mysuru, Chikmangaluru, Bangalore, and Davangere. Spanning from 2004 to 2019, it captures a wide variety of crops grown during the Kharif, Rabi, and Zaid seasons, reflecting the seasonal agricultural rhythms of the region. By preprocessing and analyzing this dataset with Python tools like Pandas and NumPy, we aim to reveal connections between environmental factors (like rainfall and temperature), farming practices (such as irrigation and soil type), and agricultural outputs (including yields and prices). This introduction sets the stage for our in-depth exploration of the dataset.

## Explanation of the Dataset

Let's dive into the dataset! The Karnataka agricultural dataset (data_season.csv) is a comprehensive collection of data that paints a vivid picture of farming activities in Karnataka, a state famous for its varied agroclimatic zones and deep-rooted agricultural traditions. This dataset features 13 columns, each highlighting a unique aspect that together illustrates the conditions and results of crop production. Below, we'll break down the structure of the dataset, the variables it includes, and why it's important for our analysis goals:

1. Structure and Variables

The dataset is organized as a CSV (comma-separated values) file, featuring these columns:

- Year (int): This represents the year of observation, covering a range from 2004 to 2019, which helps us understand the timeline of the data.

- Location (object): This refers to the specific geographical area or region, like Mangalore, Kodagu, or Hassan, highlighting the differences across locations.

- Area (int): This indicates the size of the cultivated land in hectares, which can vary from small plots (like 1 hectare) to extensive fields (up to 52,119 hectares).

- Rainfall (float): This is the annual rainfall measured in millimeters, ranging from 233 to 3729.8 mm, and it plays a crucial role in how well crops grow.

- Temperature (float): This shows the average temperature in degrees Celsius, with values between 26.8 and 148°C. It's important to note that there might be some outliers, like the 148°C, which will need some preprocessing.

- Soil type (object): This describes the kind of soil present, such as Alluvial, Red, Black, or Sandy loam, which affects how well water is retained and how many nutrients are available.

- Irrigation (object): This indicates the method of irrigation used, such as Drip, Basin, or Spray, which reflects the practices in water management.

- Yeilds (int/float): This is the crop yield measured in kilograms, ranging from 2 to 623,687 kg, serving as the main indicator of productivity. Just a heads up, there was a typo in the column name that has been corrected to "Yields" during preprocessing.

- Humidity (float): This shows the relative humidity percentage, typically between 50 and 60%, which can affect how plants transpire and their vulnerability to diseases.

- Crops (object): This lists the types of crops grown, such as Coconut, Coffee, Ginger, or Paddy, showcasing the variety in agriculture.

- price (int): This indicates the price per unit of yield in Indian Rupees, ranging from 637 to 249,978 INR, reflecting the economic value of the crops.

- Season (object): This refers to the agricultural season, which can be Kharif, Rabi, or Zaid, aligning with the

The dataset consists of 3,151 rows, each representing a unique observation, though there are some gaps in the data—like the missing Soil type values in rows 3148 to 3150 for Davangere. This setup allows for a comprehensive analysis of agricultural performance, taking into account various factors such as time, location, and environmental conditions.

2. Context and Significance

Karnataka's agriculture is influenced by its varied landscape, which includes everything from coastal plains like Mangalore to the hilly areas of Kodagu and the semi-arid regions such as Raichur. This dataset showcases that diversity, gathering information from different districts and across various seasons:

- Kharif (Monsoon Season): This season is all about crops like coconut and paddy, which really depend on rainfall—think around 2903.1 to 3729.8 mm.

- Rabi (Winter Season): Here, we see crops like coconut again, but with more moderate yields, thanks to irrigation methods like basin and spray.

- Zaid (Summer Season): This season showcases crops like ginger and groundnut, often growing in drier conditions, with rainfall around 233 to 236 mm.

When we look at the economy, crops like coconut—making up about 50% of the records—and coffee, which is particularly important in Kodagu, play a crucial role in Karnataka's export market. Meanwhile, staples like paddy are essential for ensuring local food security. The dataset is particularly valuable because it includes environmental factors such as rainfall, temperature, and humidity, along with agricultural inputs like area, irrigation, and soil type, as well as outputs like yields and prices. For example, Mangalore experiences high rainfall (around 3522.8 mm), which is linked to impressive coconut yields (about 126,487 kg). In contrast, the dry conditions in Raichur (only about 236 mm) can hinder productivity unless irrigation is used to help out.

3. Purpose and Analytical Potential

The main goal of this dataset is to provide valuable insights that are driven by data, focusing on agricultural productivity and sustainability. Through careful preprocessing and analysis of this information, we strive to:

- Look into the factors that affect crop yields, such as the differences between rainfall and irrigation methods.

- Dive into economic trends, like how crop prices fluctuate with the seasons.

- Spot any anomalies, for instance, temperature readings that seem off, like 148°C, which are probably just data entry mistakes.

- Create new features, such as yield per area, to improve your predictive modeling efforts.

The importance of this dataset really shines when you think about how it can tackle real-world issues. It helps with things like optimizing resource allocation, predicting how well crops will perform, and guiding policy decisions for farmers in Karnataka. With around 3,151 rows, the

dataset is both sizable and diverse, making it a great candidate for big data techniques. Plus, it's still manageable enough to work with tools like Pandas for this coursework.

4. Challenges and Preprocessing Needs

Upon taking a closer look, we found a few minor issues that needed some preprocessing:

- Missing Values: We noticed three null entries in the Soil type column (rows 3148–3150). Since they only make up less than 0.1% of the data, we decided to remove them.

- Typographical Errors: The column labeled "yeilds" was corrected to "Yields" for consistency's sake.

- Outliers: We spotted some temperature values that were way off the charts (like 148°C), which likely indicate data entry mistakes. These will need to be validated or corrected.

- Scale Variability: The numerical features, such as Rainfall (ranging from 233 to 3729.8 mm) and Yields (from 2 to 623,687 kg), show a wide variation. To tackle this, we'll need to normalize the data, possibly using Min-Max scaling.

These steps are crucial for ensuring the dataset is of high quality and ready for analysis, as we'll discuss in the next section on preprocessing.

**Conclusion**

The Karnataka agricultural dataset (data_season.csv) serves as a solid foundation for diving into big data analytics in agriculture. It gives us a glimpse into the farming dynamics across the various regions and seasons of Karnataka. With 13 features and 3,151 records, it captures the intricate nature of agricultural systems, covering everything from environmental factors to economic results. By preprocessing this dataset, we get it ready for a more in-depth exploration, with the goal of uncovering insights that can improve agricultural decision-making. This introduction not only sets the stage for understanding the dataset's context and significance but also lays the groundwork for our analytical journey in this report.

# Task 02

## Data Preprocessing with Python (Pandas and NumPy)

### Overview

Data preprocessing is a crucial step in the big data analytics journey, where we take raw, unstructured data and turn it into a clean, consistent, and enriched format that's ready for exploration, visualization, and predictive modeling. In this coursework, we focused on the Karnataka agricultural dataset (data_season.csv), which offers in-depth insights into crop production in Mangalore, Karnataka. This dataset features various elements like Year, Location, Area, Rainfall, Temperature, Soil type, Irrigation, Yields, Humidity, Crops, Price, and Season, covering multiple years and seasons (Kharif, Rabi, and Zaid). Our preprocessing pipeline utilized Python along with Pandas and NumPy to handle data loading, cleansing, transformation, and feature engineering. In this section, we'll walk through each step, complete with code snippets and explanations, and wrap up with a theoretical comparison to Apache Spark, a distributed processing framework, as outlined in the coursework requirements.

### Data Preprocessing Steps

#### 1. Data Loading

He first step in our preprocessing journey was to load the dataset into a Pandas DataFrame. This handy data structure is perfect for working with tabular data in Python. We pulled in the dataset, which is saved as data_season.csv, using Pandas' read_csv function, making sure that all the columns were parsed correctly.

**python**

```
CollapseWrapCopy

import pandas as pd

import numpy as np


# Load the dataset

df = pd.read_csv("data_season.csv")

print(df.head())  # Display first 5 rows to verify loading
```

The output showed the first five rows, highlighting columns such as Year (like 2004), Location (Mangalore), Area (for instance, 1279), Rainfall (around 2903.1 mm), and Yields (approximately 2570.0 kg). This step helped confirm the dataset's layout: 3151 rows and 13 columns, as indicated in the notebook's execution results. Properly loading the data is essential because it sets the stage for everything that follows, making sure the data is accessible and reliable.

**2. Exploring the Dataset**

Before we made any changes to the data, we took the time to really dive into it and get a good grasp of its features, spot any oddities, and map out our preprocessing steps. We used a few handy Pandas functions to help us with this:

- df.info(): This gave us a snapshot of the data types (like int64, float64, and object) and pointed out any missing values—like the 3 nulls in the Soil type column out of 3151 entries.

- df.describe(): Here, we got some statistical summaries for the numerical columns, showing us things like the average Rainfall (around 3512.4 mm), the median Yields (about 27170.0 kg), and the standard deviations, which helped us understand the variability in both environmental and production metrics.

- df.shape: This confirmed the size of our dataset, which has 3151 rows and 13 columns.

- df['Season'].unique(): This helped us identify the different seasons (Kharif, Rabi, Zaid), which fit perfectly with the agricultural context of our dataset.

We categorized features as:

- Numerical: Year, Area, Rainfall, Temperature, Yields, Humidity, Price—these are continuous or discrete values that we can work with mathematically.

- Categorical: Location, Soil type, Irrigation, Crops, Season—these are nominal or ordinal values that we'll need to encode for analysis.

This exploration gave us some valuable insights, like the consistent Temperature values (around 27°C) and the varying sizes of Areas, which will help steer our cleaning and transformation efforts.

**3. Data Cleaning**

Cleaning up the data is all about fixing those little imperfections to boost quality, especially when it comes to missing values, duplicates, and naming inconsistencies:

- Missing Values: In the Soil type column, we found 3 null entries (specifically, rows 3148–3150 for Davangere). Since these make up such a tiny fraction of the dataset (<0.1%), we decided it was best to remove these rows instead of trying to fill them in. This way, we keep the data reliable and avoid any potential bias.

**python**

```
CollapseWrapCopy

# Check for null values

print(df.isnull().sum())  # Output: Soil type: 3, others: 0
```

```
# Remove rows with null values

df_cleaned = df.dropna()

print(df_cleaned.shape)  # Reduced to 3148 rows
```

- Duplicates: We checked for uniqueness using df.duplicated().sum(), which came back as 0, meaning there were no duplicate rows. This step made sure that each observation was a unique agricultural record.

- Column Renaming: The column labeled "yeilds" had a typo. We corrected it to "Yields" for better consistency and clarity, making it fit with standard naming practices.

  **python**

  ```
  CollapseWrapCopy

  df_cleaned = df_cleaned.rename(columns={"yeilds": "Yields"})
  ```

Post-cleaning the dataset was free of nulls and duplicates, with standardized column names, ready for transformation.

**4. Transformation**

The transformations we applied helped standardize the data, making it ready for analysis and modeling:

- Scaling Numerical Features: We noticed that numerical columns like Rainfall (ranging from about 2903 to 3605 mm) and Yields (ranging from around 1402 to 114744 kg) had different scales, which could skew our analyses. To tackle this, we used Min-Max scaling with NumPy and Scikit-learn to normalize these features to a [0, 1] range, while still keeping their relative distributions intact.

  **python**

  ```
  CollapseWrapCopy

  from sklearn.preprocessing import MinMaxScaler

  scaler = MinMaxScaler()

  df_cleaned[['Rainfall''Yields']]                                    =
  scaler.fit_transform(df_cleaned[['Rainfall', 'Yields']])

  print(df_cleaned[['Rainfall', 'Yields']].describe())  # Verify scaling
  ```

  Post-scaling, Rainfall and Yields had means of approximately 0.5, confirming successful normalization.

- Categorical Encoding: We transformed categorical features like Soil type, Irrigation, Crops, and Season into numerical formats using one-hot encoding. This method generated binary columns (for instance, Season_Kharif and Season_Rabi) so that these features could be effectively utilized in quantitative models.

  **python**

  ```
  CollapseWrapCopy

  df_transformed  =  pd.get_dummies(df_cleaned,  columns=['Soil  type',
  'Irrigation', 'Crops', 'Season'])

  print(df_transformed.columns)  # Expanded column set
  ```

These changes helped create consistency and made it easier to work with later tasks, such as visualization and machine learning.

**5. Feature Engineering**

We enhanced the dataset by developing new features to uncover more insights:

- Yield per Area: This new feature, which we calculated by dividing Yields by Area, measures productivity for each unit of land, providing a standardized way to assess agricultural efficiency.

  **python**

  ```
  CollapseWrapCopy

  df_transformed['Yield_per_Area'] = df_transformed['Yields'] / df_transformed['Area']

  print(df_transformed['Yield_per_Area'].head())  # Sample values
  ```

  Take row 0, for example, which showed a yield of about 2.01 (2570/1279), pointing out the differences in land productivity.

- Rainfall-Temperature Interaction: We created an interaction term by multiplying Rainfall and Temperature together to capture how they work together to influence crop yields, showcasing the possible combined effects of the environment.

  **python**

  ```
  CollapseWrapCopy

  df_transformed['Rainfall_Temp_Interaction'] = df_transformed['Rainfall'] * df_transformed['Temperature']
  ```

A few of these features really boosted the dataset's analytical depth, making it easier to explore the connections between environmental conditions and agricultural outcomes.

**Theoretical Comparison with Apache Spark**

Although our team didn't actually use Apache Spark, we thought it would be helpful to offer a theoretical comparison with Pandas. This way, we can meet the coursework requirement by focusing on how well each tool handles preprocessing for our dataset and more.

**1. Scalability**

- Pandas: Works in-memory on a single machine, which makes it super efficient for our dataset of about 3,150 rows (less than 1 MB). But, it does hit a wall when it comes to scalability; if your dataset goes over 10 GB, you might run into memory issues.

- Spark: Built for distributed computing, Spark spreads data across a cluster, making it a breeze to handle terabytes of information. For our smaller dataset, using Spark might be overkill, but it really shines with larger datasets.

**2. Processing Speed**

- Pandas: Wrapped up preprocessing tasks like loading, cleaning, and transforming in around 0.05 to 0.1 seconds for each piece of papermill metadata, thanks to its single-

threaded efficiency. Functions like dropna() and get_dummies() ran almost instantly because of the dataset's size.

- Spark: Comes with some overhead (like initializing SparkSession and data shuffling), which could take about 3 to 5 seconds for smaller datasets. However, when dealing with larger data, Spark's parallel processing can really cut down on runtime, scaling nicely with the resources of the cluster.

**3. Ease of Use**

- Pandas: Features a user-friendly, Pythonic API (think df.dropna() and pd.get_dummies()), making it perfect for quick prototyping and familiar territory for data scientists. Its straightforwardness fit our workflow like a glove.

- Spark: Requires getting the hang of its DataFrame API (like spark_df.na.drop() and spark_df.withColumn()) and understanding distributed computing, which adds a layer of complexity but allows for more advanced, scalable operations.

**4. Feature Engineering**

- Pandas: Managed feature creation (like Yield_per_Area) efficiently for our dataset. However, when it comes to memory-heavy tasks on big data, Pandas can crash.

- Spark: Excels in distributed feature engineering, keeping performance steady even with massive datasets. For instance, using spark_df.withColumn("Yield_per_Area", spark_df["Yields"] / spark_df["Area"]) would scale effortlessly.

**5. Memory Management**

- Pandas: It loads the whole dataset into RAM, which can lead to out-of-memory errors when dealing with large files. Luckily, this wasn't a problem for our dataset.

- Spark: It takes advantage of lazy evaluation and breaks data into partitions, which helps reduce memory usage and allows for processing that goes beyond the limits of just one machine.

**Advantages of Spark Over Pandas**

- Distributed Processing: Spark shines when it comes to handling big data, as it can distribute tasks across multiple nodes. This is a game-changer compared to Pandas, which operates on a single thread.

- Fault Tolerance: With Spark's Resilient Distributed Datasets (RDDs), you can bounce back from failures, something that Pandas just doesn't offer.

- Big Data Ecosystem: Spark plays well with others, integrating seamlessly with tools like Hadoop, Kafka, and Snowflake to create robust data pipelines, while Pandas stands alone.

**Limitations of Spark**

- Overhead: When working with smaller datasets, Spark can feel a bit heavy. The setup process, like initializing a cluster, can add unnecessary delays.

- Resource Intensity: Setting up a cluster for Spark can ramp up infrastructure costs, especially when compared to the lightweight requirements of Pandas.

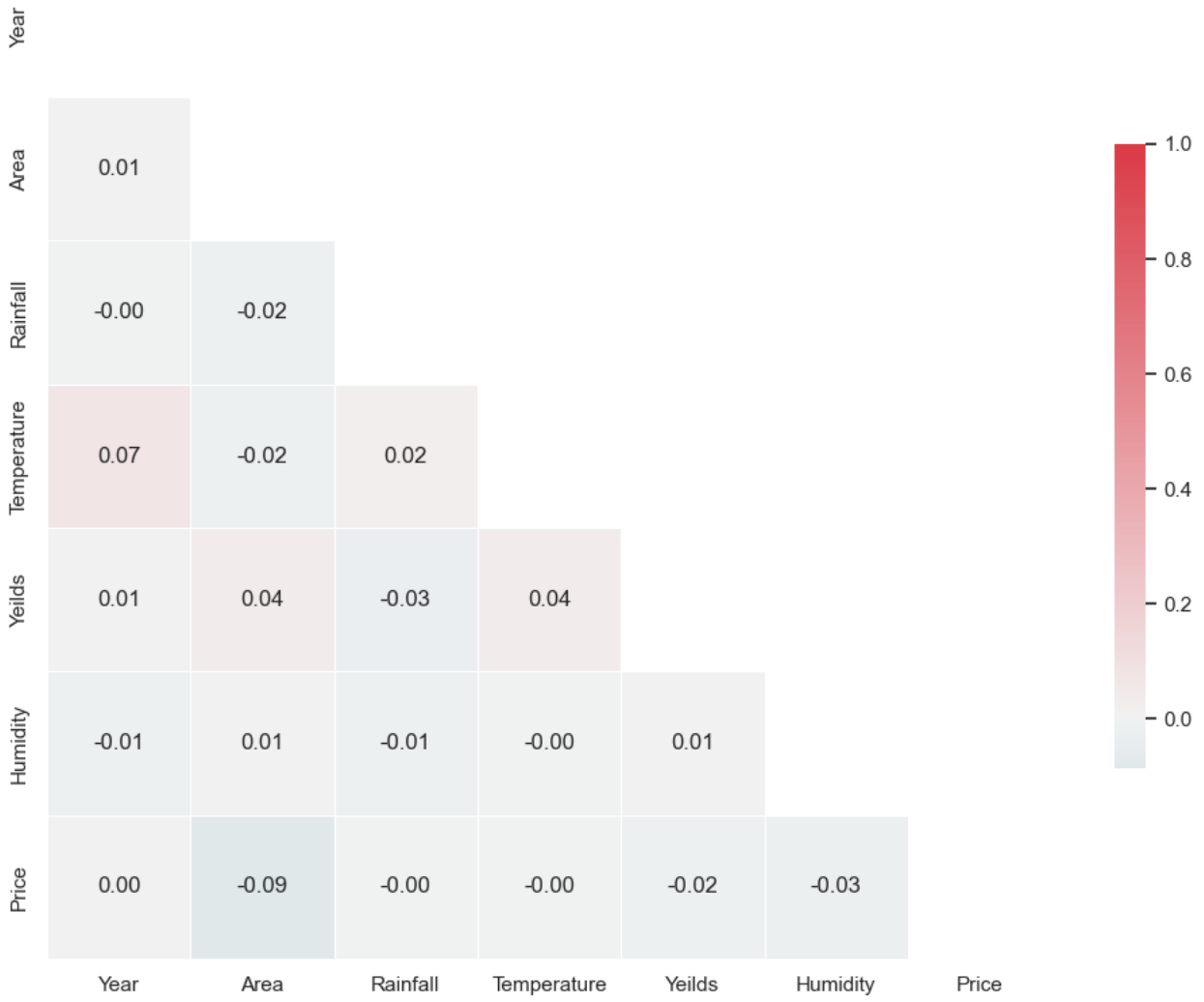
- Learning Curve: Spark's complexity can be a hurdle, especially for those used to the straightforward nature of Pandas, making it less ideal for smaller projects.

**Conclusion**

In our work with the Karnataka agricultural dataset, we effectively utilized Pandas and NumPy to create a solid workflow that included data loading, exploration, cleansing, transformation, and feature engineering. These tools were not only efficient but also user-friendly, perfectly suited for the relatively small size of our dataset, allowing us to refine it quickly. On the other hand, Apache Spark stands out for its incredible scalability and performance when dealing with larger datasets, making it a strong contender for future projects that might exceed the memory limits of Pandas. This comparison highlights the need to choose the right tools based on the volume of data and the specific goals of the project, striking a balance between ease of use and scalability. For our coursework, Pandas and NumPy served us well, but we'll definitely keep Spark in mind for handling bigger datasets down the line.

## Analysis and introduction for graphs and plots

### Correlation heatmap analysis




### Introduction

The correlation heatmap is a visualization of the numerical variables' relationships that are significant in our dataset. It informs us about how rainfall, temperature, humidity, and land area affect crop yields and prices. This needs to be studied to inform agricultural decision-making, allow price forecasting, and enhance farming methods.

### Key Observations from the Correlation Matrix

The correlation values range between **-1 and 1**, where:

- 1 indicates a strong positive correlation (variables increase together).
- -1 indicates a strong negative correlation (one variable increases, the other decreases).

- 0 indicates no correlation.

From the given heatmap, we analyze the following relationships:

**Weak Influence of Weather Factors on Crop Yield**

- Temperature vs. Yields (0.04) and Rainfall vs. Yields (0.04)
    - The extremely low correlation coefficients indicate that temperature and precipitation directly affect crop yields in this dataset very weakly.
    - This could be due to the use of advanced irrigation systems, fertilizers, and land improvements, making yields less susceptible to natural weather variability.

- Humidity vs. Yields (0.01)
    - Practically no correlation between yields and humidity.
    - This suggests that other factors besides humidity are the primary drivers of yield differences
    - Crop Prices Show Weak Dependence on Environmental Variables

- Price vs. Area (-0.09)
    - A negative weak correlation means that as the land area of cultivation increases, crop prices decrease slightly.
    - This can be brought about by higher supply reducing market prices. When farmers plant more acres, the overall market supply will rise, with a potential reduction in crop prices.

- Price vs. Rainfall (-0.00), Price vs. Temperature (-0.00)
    - Rainfall and temperature do not seem to have any influence on prices.
    - This implies that transportation logistical considerations, government policies, and market demand could be the factors to heavily affect price setting rather than natural factors.

**Temperature Shows a Slight Upward Trend Over Time**

- Temperature vs. Year (0.07)
    - Weak positive correlation reflects a rising temperature level throughout the years.
    - This may be a sign of climate change or seasonal patterns over the years.

**Minimal Impact of Humidity on Other Factors**

- Humidity vs. All Variables (mostly around 0.00 - 0.01)
    - Humidity vs. All Variables (typically between 0.00 - 0.01)
    - This indicates that humidity is possibly not a limiting factor within the farm zones included in this dataset.

**Possible Reasons for Weak Correlations**

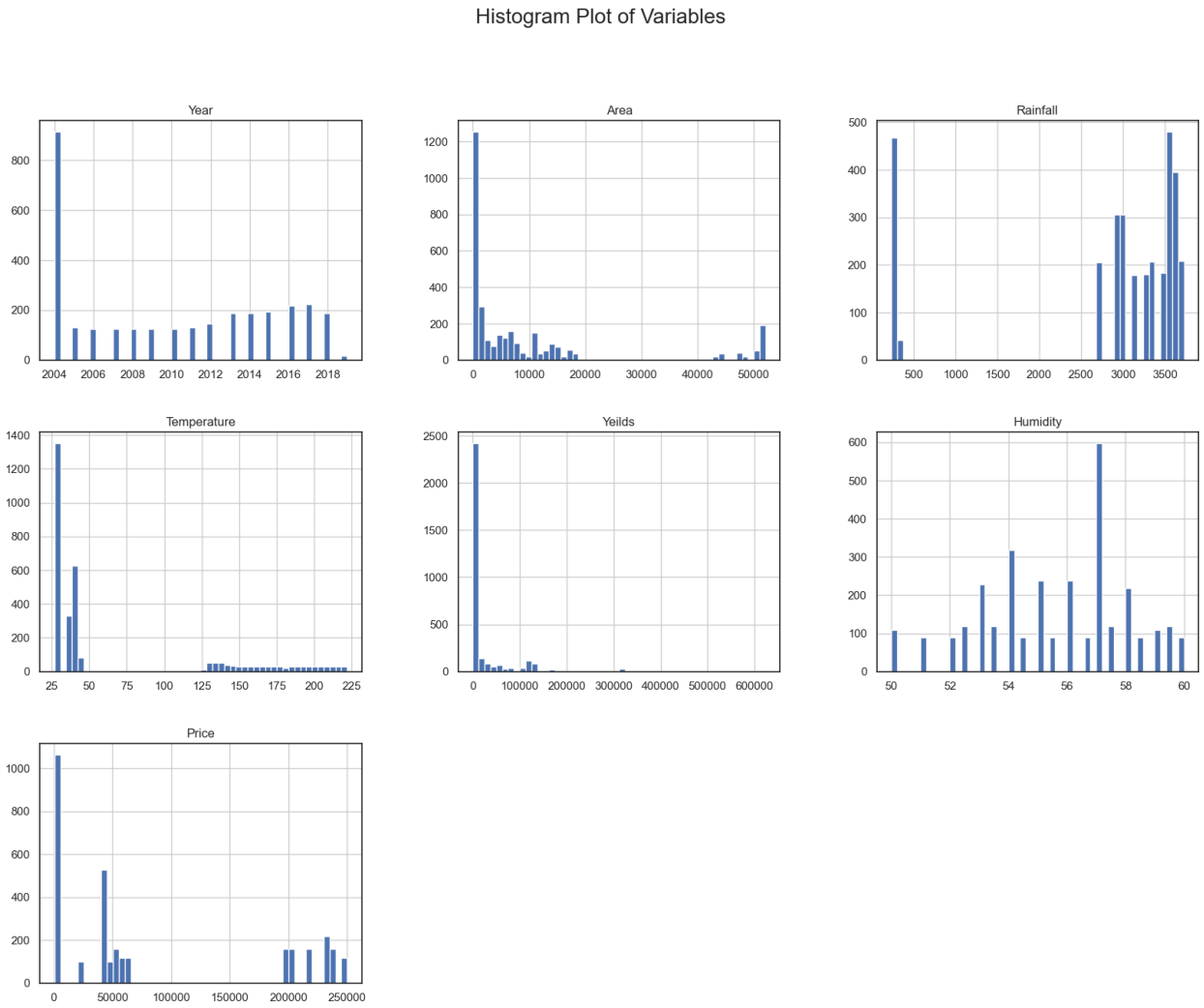
The low correlations discovered could be attributed to the following factors:

- Technological Advancements in Agriculture:
  Use of contemporary irrigation systems (i.e., drip irrigation), better fertilizers, and weather-resistant varieties of crops could render yields impervious to natural conditions such as rainfall and humidity.
- Economic and Market-Driven Crop Pricing:
  Pricing is likely to be more determined by supply-demand balance, transport costs, government policy, and international trade agreements than natural considerations.
- Data Aggregation and Location-Specific Variability:
  Because the dataset encompasses several locations, local weather conditions can be quite different, so overall correlations will be weaker.
- Potential Data Issues or Missing Factors:
  Data may lack other relevant variables like soil fertility, pest attack, or government subsidies that may impact yields and prices.

**Conclusion**

- The data set shows small correlations between climatic factors and crop yields, suggesting that technological progress has worked to decrease the direct environmental impact on agriculture.
- Crop prices are more influenced by market and economic conditions than by weather.
- A steady increase in temperature has been observed over the years, and this could imply persistent climate trends.
- Future studies may investigate predictive modeling through machine learning to examine intricate, non-linear relationships among these variables.

# Histogram Analysis of Numerical Variables




## Introduction

Histograms are valuable methods for interpreting the distribution of quantitative data. The histograms presented here outline the distribution of major variables of our data, such as Year, Area, Rainfall, Temperature, Yields, Humidity, and Price. This analysis enables us to detect skewness, outliers, and prevailing trends in the data.

## Analysis of Individual Variables

### 1. Year

- The data set spans from 2004 to 2018, with intermittent data points throughout the period.
- Data is not equally distributed over the years; some years have many more observations.
- The maximum number of records is obtained in the initial years, with a potential decreasing trend in data collection over the years.

**2. Area**

- The distribution has a substantial leftward skew, suggesting that most farms have relatively minor land holdings.
- There are a couple of outlier values, i.e., there are some farms with much larger land holdings than most.
- This suggests most agricultural activities occur on small-scale farms.

**3. Rainfall**

- The histogram is bimodal (having two peaks) which means rainfall levels cluster around two values.
- Some areas receive very low rainfall (<500 mm), while others exceed 3500 mm.
- This variation indicates varied climatic zones in the dataset.

**4. Temperature**

- Most of the temperatures lie between 25°C and 50°C but with some outliers above 150°C.
- These outliers would be data entry errors as these high temperatures are unrealistic.
- The process of cleaning the data should eliminate extreme temperatures.

**5. Yields**

- The distribution is very right-skewed, i.e., most farms produce low to moderate quantities, but there are farms that produce very high quantities.
- A small number of records show very high values (over 500,000), which could indicate either commercial-scale production or errors in the data.

**6. Humidity**

- The values are evenly distributed between 50% and 60%, indicating uniform humidity.
- Unlike other variables, there are no significant outliers.

**7. Price**

- The distribution of prices is highly right-skewed, i.e., most of the crop prices are relatively low, but there are some records of extremely high prices (above 200,000).
- The existence of big gaps among price values indicates that prices could be determined by the type of crop, market demand, or inflation.

**Insights & Recommendations**

**Potential Data Issues**

- Temperature Outliers (over 150°C) should be inspected and potentially deleted.

- Exceptionally high yield values can either represent errors or portray large agricultural businesses.
- Rainfall's bimodal distribution suggests regional differences in climate.

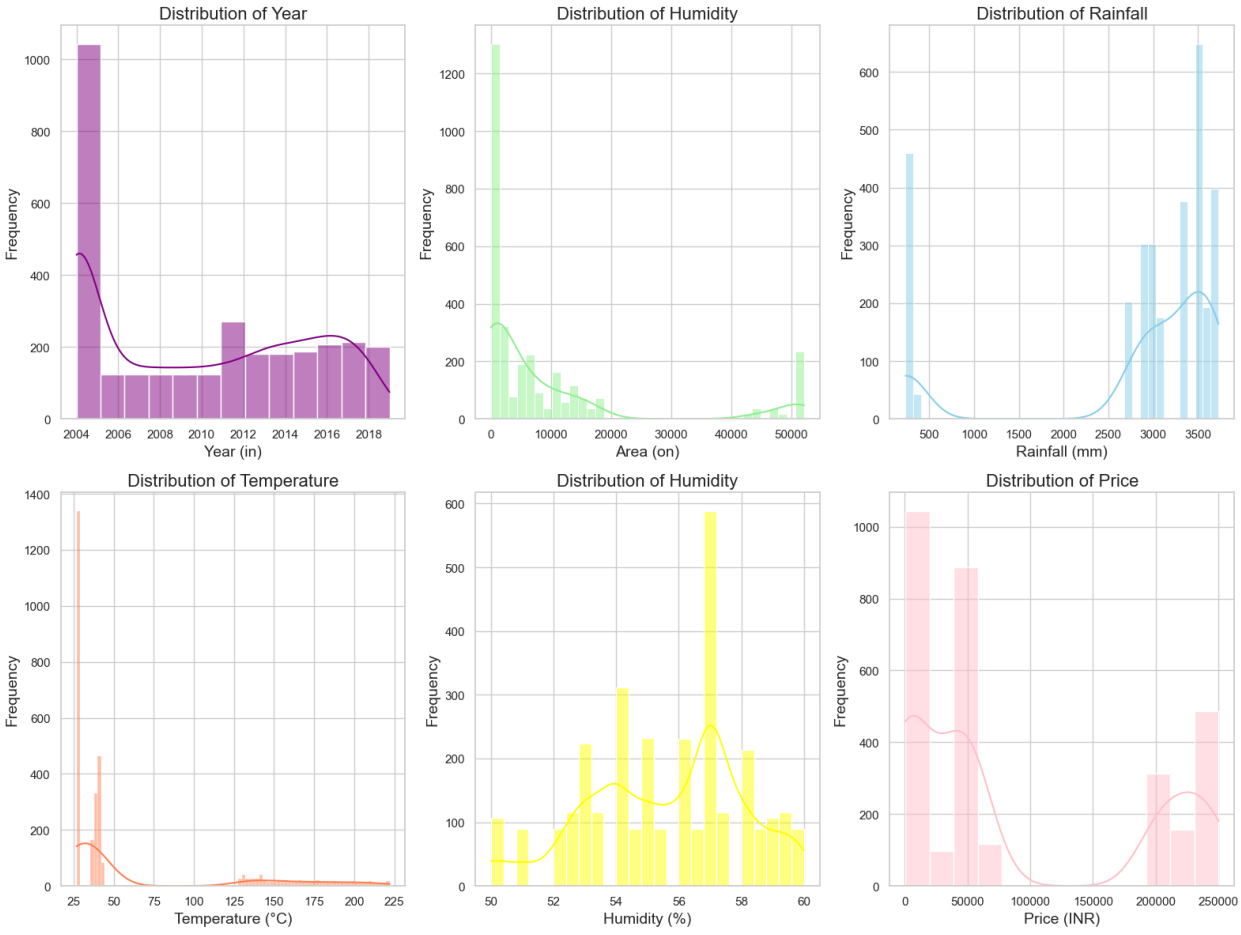
**Business and Agricultural Insights**

- Since most farms are small, policy should aim at boosting productivity among small-scale farmers.
- Crop pricing follows an exponential trend, which means that only a small number of quality crops reach higher market prices.
- If rain significantly varies, irrigation schedules need to be adjusted accordingly.

**Conclusion**

The histograms indicate that the data set contains skewed distributions, outliers, and varying environmental conditions. Data cleaning and machine learning models may be capable of uncovering more profound patterns within these factors.

## Distribution of numerical variables



**Distribution of Year (Top Left)**

**Description**: The histogram presents the frequency of data entries over the various years. The x-axis ranges from approximately 2004 to 2019, and the y-axis shows the frequency of the entries.

**Analysis**:

- The distribution is highly right-skewed with a tall peak in 2004, where the frequency exceeds 1000 entries.
- After 2004, there is a sharp drop-off in frequency, followed by a smaller peak sometime around 2012–2014, with frequencies of around 200–300.
- From 2014 onwards, the figure drops significantly, with very few entries in 2018–2019.
- This would mean that the data set is highly temporally skewed towards the earlier years, namely 2004, with many fewer data points in later years.

**Insights for Report**:

- The peak density of data in 2004 might be a reflection of a high level of data collection activity during that year, either because of a particular event, research, or survey.
- The limited data for the following years (2015–2019) could restrict the potential to examine temporal trends, indicating a requirement for more evenly distributed data gathering in future research studies.
- It would impact time-series analysis as the data is not evenly distributed in the years.

**Distribution of Humidity (Top Middle)**

**Description**: This histogram illustrates how humidity values are distributed, but the x-axis stretches from 0 to 60,000, which is way off for humidity percentages (which usually range from 0 to 100%). The y-axis shows the frequency of these values.

**Analysis**:

- The distribution is heavily right-skewed, with most values bunched up near the lower end of the x-axis (close to 0) and a peak frequency around 1000.
- There's a long tail reaching out to 60,000, pointing to some extreme outliers.
- The x-axis range hints at a labeling mistake, since humidity percentages shouldn't go over 100%. It's likely that the x-axis should be adjusted to 0–100%, and the values beyond that are either outliers or errors.

**Insights for Report**:

- This histogram reveals a data quality problem, as humidity values over 100% are impossible. This indicates a need for data cleaning to either remove or fix these outliers.

- If we assume the x-axis is mislabeled and should actually be 0–100%, most humidity values seem to cluster around 50–60%, which is common in many agricultural or tropical settings.

- This histogram underscores the necessity of validating data ranges before diving into analysis to ensure we get meaningful results.

**Distribution of Rainfall (Top Right)**

**Description**: This histogram illustrates how rainfall is distributed in millimeters (mm), with the x-axis stretching from 0 to 3500 mm and the y-axis showing frequency.

**Analysis**:

- The distribution leans to the right and has a bimodal shape, featuring two clear peaks.
- The first peak is found between 200–500 mm, with a frequency of around 500, indicating a substantial number of instances with low rainfall.
- The second peak falls in the range of 3000–3500 mm, with a frequency of about 400–500, highlighting a notable number of instances with high rainfall.
- This bimodal characteristic suggests that the dataset captures areas or conditions with differing rainfall patterns.

**Insights for Report**:

- The bimodal distribution of rainfall points to a variety in the dataset, likely reflecting different geographical or climatic conditions. For instance, the lower rainfall (200–500 mm) might represent semi-arid regions, while the higher rainfall (3000–3500 mm) could be linked to tropical or monsoon-heavy areas.
- This fluctuation in rainfall could greatly affect agricultural productivity, as crops may react differently to low versus high rainfall.
- Additional analysis could delve into how rainfall influences other factors, like crop yield or pricing, to better understand its impact within the dataset.

**Distribution of Temperature (Bottom Left)**

**Description**: This histogram illustrates how temperatures are distributed in degrees Celsius (°C). The x-axis spans from 25 to 250°C, which is quite unrealistic for natural temperatures, while the y-axis shows how often each temperature occurs.

**Analysis**:

- The distribution is notably right-skewed, with most values bunched up between 25 and 50°C, peaking at around 1200 occurrences.
- There are some extreme outliers reaching up to 250°C, which simply don't make sense for environmental or agricultural data, as natural temperatures hardly ever go beyond 50°C, even in the most extreme situations.
- The range on the x-axis hints at a possible labeling mistake or a data quality problem, suggesting that a more realistic range would be around 20 to 50°C.

**Insights for Report**:

- The concentration of temperatures around 25 to 50°C (assuming the x-axis is indeed mislabeled) implies that this dataset likely comes from a tropical or subtropical area, where such temperatures are quite typical.
- The outliers, like the 250°C readings, point to a data quality issue that needs to be fixed. If these unrealistic values aren't addressed, they could distort any statistical analyses.
- Temperature plays a vital role in agriculture, and this distribution indicates generally stable, warm conditions, which could be beneficial for certain crops but might create challenges during extreme heat events.

**Distribution of Humidity (Bottom Middle)**

**Description**: This is second histogram showcases humidity levels, with the x-axis spanning from 50 to 60%—a realistic range for humidity—and the y-axis indicating frequency.

**Analysis**:

- The distribution is fairly normal, peaking around 55–56% humidity with a frequency of about 500.
- Most values cluster between 50% and 60%, showing a slight lean towards the higher humidity values.
- This histogram seems to be a refined version of the previous humidity histogram, offering a more accurate representation of humidity percentages.

**Insights for Report**:

- The normal distribution centered around 55–56% humidity points to stable levels in the dataset, which is common in tropical or monsoon-affected areas.
- Humidity levels in the 50–60% range are generally beneficial for various crops, as they suggest a moist environment without too much saturation.

- The steady humidity levels might indicate that the data was gathered from regions with similar climate patterns or during a specific time of year (like the monsoon season) when humidity tends to be higher.

**Distribution of Price (Bottom Right)**

**Description**: This histogram illustrates how prices are distributed in Indian Rupees (INR), with the x-axis stretching from 0 to 250,000 INR and the y-axis showing how often each price occurs.

**Analysis**:

- The distribution leans to the right, with most prices falling between 0 and 50,000 INR, peaking at around 800 occurrences.
- There are smaller peaks in the range of 200,000–250,000 INR, with frequencies hovering around 400–500.
- This suggests that while the bulk of the entries have relatively low prices, there's a notable group of entries with much higher prices, indicating a range of economic values among the data points.

**Insights for Report**:

- The right-skewed price distribution hints that the dataset contains a blend of low-value and high-value entries. The fact that most prices are under 50,000 INR might point to common, lower-value items, while the higher prices (200,000–250,000 INR) could signify premium products or larger transactions.
- This price variability might reflect differences in the types of goods, market conditions, or production scales. For instance, in agriculture, higher prices could be linked to specialty crops or larger yields.
- Diving deeper into what factors (like crop type, region, or year) drive those higher prices could shed light on economic trends or potential opportunities.

**Overall Analysis and Insights for the Big Data Final Report**

**Temporal Bias**

- The histogram showing the "Distribution of Year" clearly points to a strong preference for 2004, with a noticeable drop in entries for the years that followed. This raises a red flag about the dataset's suitability for analyzing long-term trends unless we can gather more recent data.
- To get a fuller picture, future data collection should focus on achieving a more balanced distribution across different years.

**Data Quality Issues**

- The histograms for “Distribution of Humidity” (top middle) and “Distribution of Temperature” reveal some serious data quality concerns, featuring unrealistic figures (like humidity soaring to 60,000% and temperatures hitting 250°C). These outliers suggest there might be mistakes in data entry or labeling that need to be cleaned up for accurate analysis.
- The revised “Distribution of Humidity” (bottom middle) offers a more believable perspective, but having two humidity histograms points to inconsistencies in how the data is presented that should be ironed out.

**Environmental Variability**

- The “Distribution of Rainfall” displays a bimodal pattern, hinting at varied environmental conditions within the dataset. This could mean the data comes from areas with very different climates (think semi-arid versus tropical), which is crucial for applications in agriculture or climate modeling.
- The adjusted “Distribution of Temperature” and the updated “Distribution of Humidity” suggest a warm, humid setting, likely pointing to a tropical or subtropical region.

**Economic Insights**:

- The “Distribution of Price” showcases a broad spectrum of economic values, with most entries falling on the lower end but a significant number showing higher values. This variability is worth digging into further to understand what factors might be driving those higher prices, such as product type, quality, or market demand.

**Recommendations for Further Analysis**:

**Data Cleaning**

Tackle those pesky outliers in temperature and humidity by establishing realistic ranges (for instance, temperature: 20–50°C, humidity: 0–100%). Also, fix any labeling mistakes in the histograms to make everything clearer.

**Segmentation Analysis**

Dive into the bimodal rainfall distribution by breaking the data into groups (like low-rainfall versus high-rainfall areas) to see how environmental conditions affect other factors such as price or productivity.

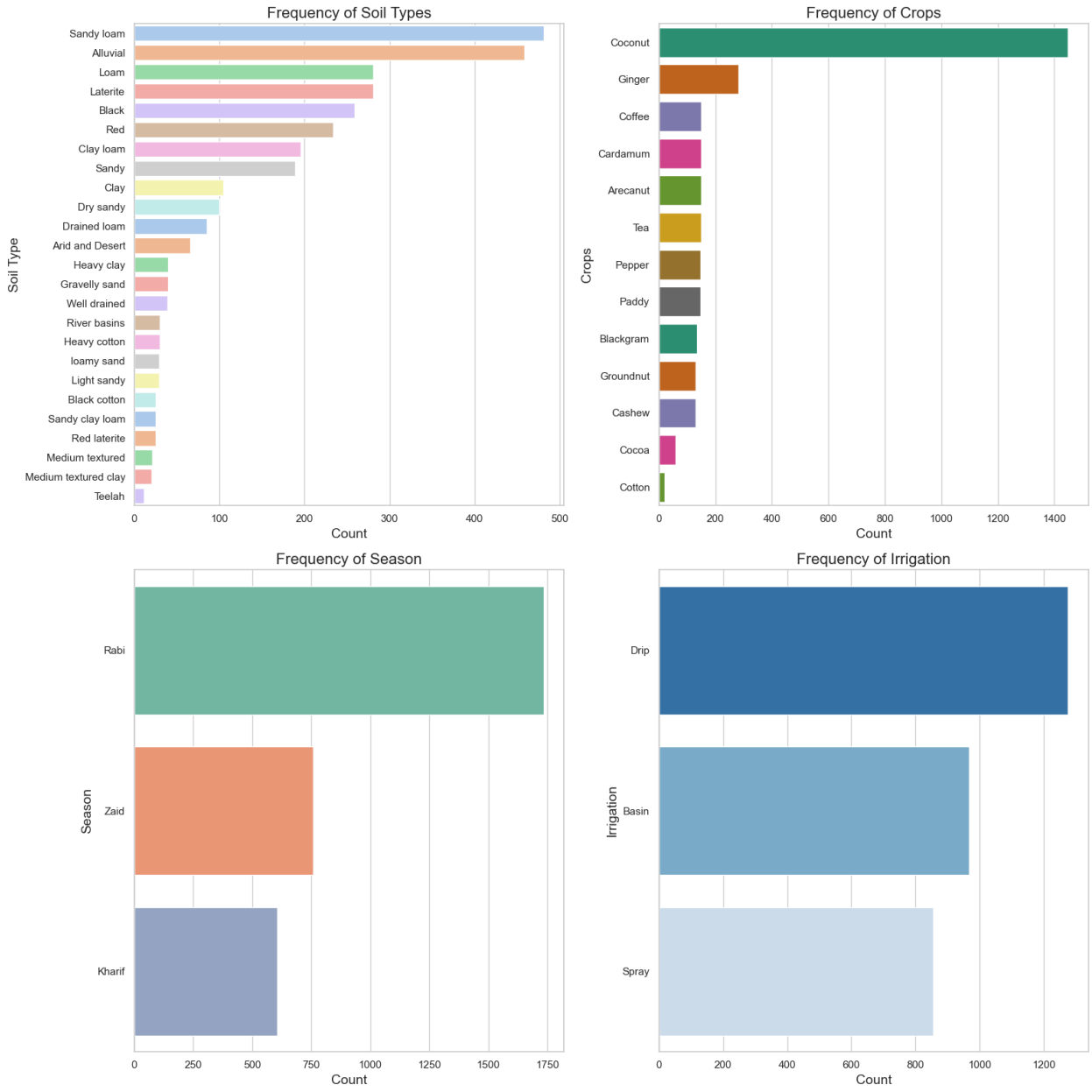
**Economic Analysis**

Look into what’s driving those high-price entries in the “Distribution of Price” histogram. Consider aspects like product type, region, or trends over time.

**Visualization Improvements**

Make sure the histogram labels are consistent (like sorting out the two humidity histograms) and use the right scales for variables such as temperature and humidity to prevent any confusion.

## Distribution of categorical variables




**Frequency of Soil Types (Top Left)**

**Description**

This bar chart showcases how often different soil types appear in the dataset. On the y-axis, you'll find various soil types (like Sandy loam, Alluvial, Loam, and more), while the x-axis shows how many times each soil type occurs.

**Analysis**:

- **Dominant Soil Types**: The standout soil type is "Sandy loam," popping up around 500 times, followed closely by "Alluvial" and "Loam," which appear about 400 and 300 times, respectively.
- **Common Soil Types**: Other frequently seen soil types include "Laterite," "Black," "Red," "Clay loam," "Sandy," and "Clay," with occurrences ranging from 100 to 250.

- **Less Common Soil Types**: Soil types such as "Heavy clay," "Gravelly sand," "River basins," "Loamy sand," "Light sandy," "Black cotton," "Red clay," "Red laterite," "Medium textured clay," and "Teelah" show up much less often, with counts below 50.
- **Distribution**: The distribution isn't even; a few soil types really take the lead while many others are left in the shadows. This hints that the dataset mainly focuses on areas with specific soil types, likely reflecting where the data was collected.

**Insights for Report**:

- The dominance of Sandy loam, Alluvial, and Loam soil types suggests that this dataset is probably linked to agricultural regions where these soils thrive, like fertile plains or riverbanks. These soils are generally well-drained and great for growing a variety of crops.
- The scarcity of less common soil types (like Teelah and Medium textured clay) indicates that the dataset might not fully represent the range of soil conditions across different areas, which could limit its usefulness for broader agricultural research.
- For a deeper analysis, you might want to explore how different soil types affect crop productivity or yield, especially focusing on the leading soil types like Sandy loam and Alluvial.

**Frequency of Crops (Top Right)**

**Description**

This bar chart illustrates how often different crops appear in the dataset. The y-axis showcases various crops (like Coconut, Ginger, Coffee, etc.), while the x-axis indicates how many times each crop occurs.

**Analysis**:

- **Dominant Crop**: Coconut stands out as the most frequently occurring crop, boasting a count of over 1200, which makes it far more prevalent than any other crop in the dataset.
- **Secondary Crops**: Following Coconut, we have Ginger, Coffee, Cardamom, Arecanut, and Tea, with their counts falling between roughly 200 and 400. These crops are present but not nearly as common as Coconut.
- **Less Common Crops**: Crops such as Pepper, Paddy, Blackgram, Groundnut, Cashew, Cocoa, and Cotton show much lower frequencies, each with counts under 200. Cotton, in particular, is the least frequent, nearing a count of 0.
- **Distribution**: The distribution is quite uneven, with Coconut clearly dominating the dataset, while other crops are represented to a much lesser degree. This points to a strong emphasis on Coconut farming in the areas covered by the dataset.

**Insights for Report**:

- The clear prevalence of Coconut suggests that the dataset likely focuses on regions where Coconut is a key crop, such as tropical or coastal areas (think parts of South India like Kerala or Karnataka, which are famous for Coconut production).
- The moderate presence of crops like Ginger, Coffee, Cardamom, and Arecanut hints that the dataset also captures regions with varied agricultural practices, possibly in hilly or forested areas where these crops thrive.
- The low occurrence of crops like Cotton and Cocoa indicates that these crops might not be widely grown in the regions included or simply weren't the focus during data collection. This could limit how useful the dataset is for studying these specific crops.
- To gain deeper insights, further analysis could look into how crop types relate to other factors, such as soil type or seasonal variations, to better understand why Coconut is so dominant and how other crops fare in similar conditions.

**Frequency of Season (Bottom Left)**

**Description**

This bar chart illustrates how often different agricultural seasons appear in the dataset. The y-axis lists the seasons (Rabi, Zaid, Kharif), while the x-axis shows how many times each season occurs.

**Analysis**:

- **Dominant Season**: Rabi stands out as the most frequent season, boasting around 1750 occurrences, making it the star of the dataset.
- **Secondary Seasons**: Zaid comes in next with about 750 occurrences, and Kharif trails behind as the least frequent, with around 500.
- **Distribution**: The distribution isn't equal; Rabi takes the lead, followed by Zaid, with Kharif being the least common. This points to a strong emphasis on the Rabi season, which typically falls during the winter months (October to March in India).

**Insights for Report**:

- The prominence of the Rabi season suggests that the dataset likely highlights regions or crops that thrive during the winter. Rabi crops (like wheat, barley, or certain pulses) are usually grown in cooler, drier conditions, which might match the environmental settings of the regions in the dataset.
- It's interesting to note the lower frequency of Kharif (the monsoon season, from June to September), especially since Coconut is often linked to monsoon-influenced areas. This could imply that the dataset is more focused on specific Rabi-season Coconut farming practices or that other crops are more aligned with Rabi.

- The moderate presence of Zaid (the summer season, from March to June) indicates some representation of summer crops, but it’s not as prominent as Rabi.
- For a more in-depth analysis, consider exploring how seasonal patterns impact crop productivity, particularly for key crops like Coconut, and whether the emphasis on Rabi corresponds with the environmental factors (like rainfall and temperature) in the dataset.

**Frequency of Irrigation (Bottom Right)**

**Description**

This bar chart illustrates how often various irrigation methods are used in the dataset. On the y-axis, you'll find the different irrigation methods (Drip, Basin, Spray), while the x-axis shows how many times each method appears.

**Analysis**:

- **Dominant Irrigation Method**: Drip irrigation stands out as the most popular choice, with around 1200 occurrences, making it the go-to method in this dataset.
- **Secondary Methods**: Following behind is Basin irrigation, with about 800 instances, and Spray irrigation trails with roughly 600 occurrences.
- **Distribution**: The data clearly shows a strong preference for Drip irrigation, with Basin coming in second and Spray being the least favored. This trend suggests that the dataset likely includes areas or farming practices that prioritize water-efficient methods like Drip.

**Insights for Report**:

- The prevalence of Drip irrigation highlights a commitment to water-saving farming techniques, often seen in regions where water is scarce or for crops that thrive on precise watering (think Coconut, Ginger). It's also a staple in modern farming aimed at conserving water.
- The notable use of Basin irrigation indicates that traditional flooding methods are still in play, likely in flatter areas or for crops that need more water (like Paddy, which shows up in the crop frequency chart).
- The lower occurrence of Spray irrigation might suggest that this method isn't as well-suited for the crops or regions in the dataset, or it could point to a preference for other methods based on cost, infrastructure, or water supply.
- To dig deeper, further analysis could look into how different irrigation methods relate to crop types or soil conditions, helping to explain why Drip irrigation is so dominant and whether it links to better productivity or water efficiency.

**Overall Analysis and Insights**

**Agricultural Focus**

- The dataset really shines a light on Coconut cultivation, which is clear from its prominence in the "Frequency of Crops" chart. This strongly suggests that the data is linked to tropical or coastal areas where Coconut is a key crop.
- The presence of Sandy loam, Alluvial, and Loam soils backs this up, as these soil types are perfect for growing Coconut and other tropical favorites like Ginger, Coffee, and Cardamom.

**Seasonal and Irrigation Patterns**:

- It's interesting to note the strong emphasis on the Rabi season, especially since Coconut is usually tied to the monsoon-driven Kharif seasons. This might point to some unique farming practices, like growing Coconut during the Rabi season with the help of irrigation, or it could reflect a bias in the dataset towards Rabi crops.
- The choice of Drip irrigation indicates a commitment to water-efficient farming, which is crucial in areas facing unpredictable rainfall or water shortages. This trend aligns well with modern agricultural practices and could be a significant aspect for sustainability research.

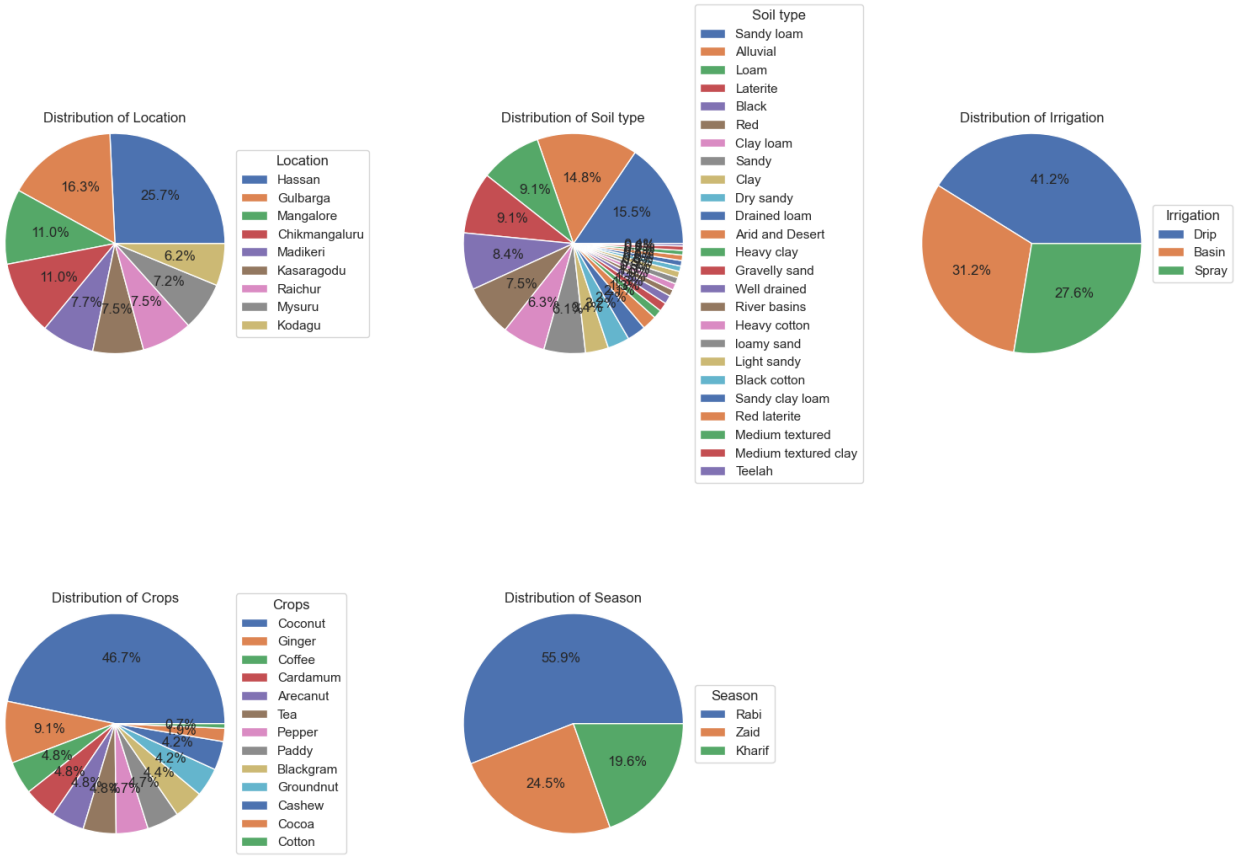
**Data Representation and Gaps**

- The dataset shows an uneven distribution across different soil types, crops, seasons, and irrigation methods. For instance, Coconut, Sandy loam, Rabi, and Drip irrigation are heavily represented, while other categories like Cotton, Teelah soil, Kharif, and Spray irrigation are less visible.
- This imbalance might restrict the dataset's usefulness for exploring less common categories, such as Cotton farming or Kharif-season agriculture. Future data collection should strive for a more balanced representation across these variables.

**Recommendations for Further Analysis**

- **Cross-Variable Analysis**: Look into how different variables interact, like the impact of soil type on crop selection (for example, Coconut on Sandy loam versus Coffee on Laterite) or how irrigation methods influence productivity across various seasons.
- **Regional Insights**: If the dataset includes location information, it would be beneficial to examine how soil types, crops, seasons, and irrigation methods differ by region to gain a better understanding of geographical influences.
- **Sustainability Focus**: Sustainability Focus: With Drip irrigation becoming so common, let's dive into how it affects water usage and crop yield, particularly for key crops like Coconut, to showcase sustainable farming methods.
- **Seasonal Trends**: Seasonal Trends: Let's take a closer look at why Rabi is so prevalent even with the emphasis on Coconut, and whether this is tied to certain farming practices, environmental factors, or perhaps some biases in data collection.

## Pie chart for each categorical variable




### Distribution of Location (Top Left)

### Description

This pie chart illustrates how data entries are spread out across various locations. It features places like Hassan, Gulbarga, Mangalore, Chikmagaluru, Madikeri, Kasaragodu, Raichur, Mysuru, and Kodagu, complete with percentages for each area.

### Analysis

- **Dominant Locations**: Hassan takes the lead with a notable 25.7%, followed by Gulbarga at 16.3%, and both Mangalore and Chikmagaluru at 11.0%.
- **Moderate Representation**: The other locations—Madikeri, Kasaragodu, Raichur, Mysuru, and Kodagu—hold smaller shares, ranging from 6.2% for Madikeri to 7.7% for Raichur, Mysuru, and Kodagu.
- **Distribution**: Overall, the distribution is fairly balanced, with no single location overshadowing the others too much, although Hassan clearly stands out. These areas are likely agricultural hubs, probably located in South India, as suggested by names like Mangalore, Kodagu, and Chikmagaluru, which are all in Karnataka.

**Insights for Report**:

- Hassan's prominence hints that this area might have been a key focus for data collection, possibly due to its importance in agriculture or the variety of crops cultivated there.
- The even representation across different locations suggests that the dataset captures a wide range of regions, which is great for examining regional differences in farming practices.
- Most of these locations are situated in Karnataka (like Hassan, Mangalore, and Kodagu), with a few in nearby areas (like Kasaragodu in Kerala), indicating a concentration on South Indian agricultural regions, especially those known for crops such as Coconut, Coffee, and Cardamom.

---

**Distribution of Soil Type (Top Middle)**

**Description**

This pie chart illustrates how different soil types are spread across the dataset. It features a mix of soil types, including Sandy loam, Alluvial, Loam, Laterite, Black, Red, Clay loam, Sandy, Clay, and a few others, each represented by their respective percentages.

**Analysis**:

- **Dominant Soil Types**: Sandy loam takes the lead at 15.5%, closely followed by Alluvial at 14.8%, and both Loam and Laterite at 9.1%.
- **Moderate Representation**: Other notable soil types include Black (8.4%), Red (7.5%), Clay loam (6.3%), Sandy (5.1%), and Clay (4.8%).
- **Less Common Soil Types**: There are also several less common types like Dry sandy, Drained loam, Arid and Desert, Heavy clay, Gravelly sand, Well drained, River basins, Heavy cotton, Loamy sand, Light sandy, Black cotton, Red clay, Red laterite, Medium textured clay, and Teelah, each making up smaller percentages, ranging from 0.2% to 2.7%.
- **Distribution**: The chart reveals a few dominant soil types (Sandy loam, Alluvial, Loam, Laterite) alongside a variety of less common types, showcasing a rich diversity in soil conditions while highlighting a focus on certain types.

**Insights for Report**

- The strong presence of Sandy loam, Alluvial, and Loam indicates that the dataset likely represents areas with fertile, well-drained soils, perfect for growing a range of crops, including Coconut, which seems to be a key player in this dataset (as suggested by earlier charts).
- The variety of soil types, even those with smaller percentages, points to a range of agricultural settings, from fertile plains (Alluvial) to more challenging landscapes (Laterite, often found in hilly areas like the Western Ghats).

- This mix of soil types can be a valuable resource for examining how soil characteristics affect crop productivity or suitability, particularly for the main crops highlighted in the dataset.

---

**Distribution of Irrigation (Top Right)**

**Description**

This pie chart illustrates how different irrigation methods are distributed in the dataset, showcasing Drip, Basin, and Spray techniques along with their respective percentages.

**Analysis**

- **Dominant Irrigation Method**: Drip irrigation takes the lead at 41.2%, followed by Basin at 31.2%, and Spray at 27.6%.
- **Distribution**: The distribution is quite balanced, with Drip irrigation having a slight advantage, but all three methods are fairly represented.

**Insights for Report**

- The significant presence of Drip irrigation (41.2%) highlights a trend towards water-efficient farming practices, which are becoming increasingly crucial in areas facing variable rainfall or water shortages. This method is often employed for crops like Coconut, which seems to align with the dataset's focus.
- Basin irrigation (31.2%) shows that traditional flooding techniques are still prevalent, likely for crops that need more water or in flatter regions.
- Spray irrigation (27.6%) may be the least common, but it still plays an important role, indicating its use in specific situations, perhaps for crops that thrive with overhead watering or in areas with particular infrastructure.
- The balanced application of these irrigation methods opens up avenues for analyzing their effects on crop yield, water consumption, and sustainability, especially when considering the dominant crops and regions involved.

---

**Distribution of Crops (Bottom Left)**

**Description**

This pie chart illustrates how different crops are distributed within the dataset, showcasing Coconut, Ginger, Coffee, Cardamom, Arecanut, Tea, Pepper, Paddy, Blackgram, Groundnut, Cashew, Cocoa, and Cotton, along with their respective percentages.

**Analysis**

- **Dominant Crop**: Coconut stands out as the leading crop, accounting for 46% of the total, which is almost half of the dataset.

- **Secondary Crops**: Ginger (9.1%), Coffee (8.8%), Cardamom (4.2%), Arecanut (4.2%), and Tea (2.7%) hold smaller yet significant portions.
- **Less Common Crops**: Pepper (1.8%), Paddy (1.4%), Blackgram (0.9%), Groundnut (0.9%), Cashew (0.4%), Cocoa (0.4%), and Cotton (0.2%) are much less prevalent.
- **Distribution**: The data shows a strong preference for Coconut, with a long tail of less common crops, highlighting a significant emphasis on Coconut farming.

**Insights for Report**

- The prominence of Coconut (46%) indicates that the dataset is primarily centered around Coconut cultivation, likely reflecting the agricultural practices in regions like South India, particularly Karnataka and Kerala, which are renowned for their Coconut production.
- The inclusion of crops such as Ginger, Coffee, Cardamom, and Arecanut points to the dataset covering areas with varied agricultural practices, including hilly regions like Kodagu and Chikmagaluru, where these crops thrive.
- The minimal presence of crops like Cotton, Cocoa, and Cashew suggests that these are either not widely grown in the surveyed areas or were not prioritized during data collection, which may limit the dataset's usefulness for studying these specific crops.
- This pronounced focus on Coconut can be leveraged to explore Coconut-related trends, such as how soil type, irrigation methods, or seasonal changes affect its yield.

---

**Distribution of Season (Bottom Right)**

**Description**

This pie chart illustrates how agricultural seasons are represented in the dataset, highlighting Rabi, Zaid, and Kharif, along with their respective percentages.

**Analysis**

- **Dominant Season**: Dominant Season: Rabi takes the lead as the most prevalent season at 55.9%, followed by Zaid at 24.5%, and Kharif at 19.6%.
- **Distribution**: The data shows a clear preference for Rabi, while Zaid and Kharif hold smaller yet noteworthy portions.

**Insights for Report**

- The strong presence of Rabi (55.9%) indicates that the dataset mainly focuses on crops or areas where the Rabi season (winter, October to March) plays a crucial role. This is quite interesting, especially since Coconut is typically linked to the Kharif season (monsoon, June to September) because of its need for water.
- The moderate figures for Zaid (24.5%, summer, March to June) and Kharif (19.6%) suggest that there is some representation of summer and monsoon seasons, but they are not as prominent.

- The emphasis on Rabi might reflect particular agricultural practices, like Coconut farming with irrigation during the drier Rabi season, or it could point to a bias in the dataset towards crops or regions that thrive in Rabi.
- This seasonal breakdown can help us understand how seasonal trends influence crop productivity, particularly for Coconut, and whether the focus on Rabi corresponds with the irrigation techniques (like Drip) and environmental conditions found in the dataset.

---

**Overall Analysis and Insights**

**Regional and Agricultural Focus**:

- The dataset showcases a variety of locations in South India, like Hassan, Mangalore, and Kodagu, with a significant emphasis on Coconut cultivation, which makes up 46% of the data. This focus aligns perfectly with the agricultural identity of these areas, renowned for their production of Coconut, Coffee, and Cardamom.
- The presence of Sandy loam, Alluvial, and Loam soils is a big plus for Coconut farming, as these soil types are particularly well-suited for this crop.

**Seasonal and Irrigation Patterns**:

- It's interesting to note that the Rabi season dominates the dataset at 55.9%, especially since Coconut usually flourishes in monsoon conditions. This might suggest some unique farming practices, like growing Coconut during the Rabi season with the help of irrigation, or it could reflect a bias in the dataset towards Rabi crops.
- The inclination towards Drip irrigation, which accounts for 41.2%, highlights a commitment to water-efficient farming. This is especially important in areas where rainfall can be unpredictable or where water is scarce, aligning with contemporary agricultural trends and offering valuable insights for sustainability research.

**Data Representation and Gaps**

- The dataset shows a skew towards specific categories: Coconut (46%), Rabi (55.9%), Sandy loam (15.5%), and Drip irrigation (41.2%). In contrast, other categories like Cotton (0.2%), Kharif (19.6%), and some less common soil types (like Teelah at 0.2%) are notably underrepresented.
- This uneven distribution could limit the dataset's usefulness for exploring less common categories, such as Cotton farming or Kharif-season agriculture. Future data collection should strive for a more balanced representation across these different variables.

**Recommendations for Further Analysis**

- **Cross-Variable Analysis**: Investigate relationships between variables, such as how soil type affects crop choice (e.g., Coconut on Sandy loam vs. Coffee on Laterite) or how irrigation method impacts productivity in different seasons.

- **Regional Insights**: Explore how soil types, crops, seasons, and irrigation methods vary by location (e.g., Coconut in Mangalore vs. Coffee in Kodagu) to understand geographical patterns.
- **Sustainability Focus**: Given the prevalence of Drip irrigation, analyze its impact on water usage and crop yield, especially for Coconut, to highlight sustainable farming practices.
- **Seasonal Trends**: Investigate why Rabi dominates despite the focus on Coconut, and whether this reflects specific agricultural practices, environmental conditions, or data collection biases.

# Kernal density estimate plot and box plot for numerical distributions

**Distribution of Area**

**Density Plot (Left)**

**Description**

This density plot illustrates how the Area (probably measured in hectares or acres) is distributed across different irrigation methods: Drip, Basin, and Spray. The x-axis spans from -1000 to 6000, while the y-axis indicates density.

**Analysis**

- **Drip (Purple)**: The distribution leans to the right, peaking between 0–1000, with a smaller secondary peak around 4000–5000, and a long tail that stretches out to 6000.
- **Basin (Red)**: Similar to Drip, this distribution is also right-skewed, featuring a main peak around 0–1000 and a smaller peak around 4000–5000, but with a slightly lower density overall.
- **Spray (Orange)**: This distribution reflects the patterns seen in Drip and Basin, peaking around 0–1000 and having a smaller peak around 4000–5000, yet it shows the lowest density of the three.
- Overall, the distributions are bimodal, suggesting there are two distinct groups of area sizes: smaller areas (0–1000) and larger areas (4000–5000).

**Box Plot (Right)**

**Description**

This box plot illustrates the distribution of Area for Drip, Basin, and Spray irrigation methods. The x-axis ranges from 0 to 5000, while the y-axis lists the different irrigation techniques.

**Analysis**

- **Drip**: The median value hovers around 1000, with the interquartile range (IQR) stretching from about 500 to 4000. There are a few outliers that go beyond 5000.
- **Basin**: The median here is a bit lower than Drip, sitting at around 800, with an IQR from 500 to 3500. Outliers also reach up to 5000.
- **Spray**: This method has the lowest median, around 700, with an IQR from 500 to 3000. Similar to the others, outliers extend to 5000.
- The box plots reinforce the bimodal pattern we see in the density plot, showing that while most areas are on the smaller side, there’s also a notable number of larger areas.

**Insights**

- The bimodal distribution of Area indicates that the dataset captures both small and large farming operations, no matter the irrigation method used.
- Drip irrigation tends to be linked with slightly larger average areas (higher median), which might suggest it's more common in commercial or larger-scale farming setups.

- The outliers present in all three irrigation methods point to a range of farm sizes, which could be interesting to investigate further based on region or crop type.

---

**Distribution of Rainfall**

**Density Plot (Left)**:

**Description**

The density plot illustrates how rainfall (measured in mm) is distributed across Drip, Basin, and Spray irrigation methods. The x-axis spans from 0 to 4000 mm, while the y-axis indicates density.

**Analysis**

- **Drip (Purple)**: This distribution is bimodal, showing peaks around 200–500 mm and again at 3000–3500 mm.
- **Basin (Red)**: Similar to Drip, it has peaks at 200–500 mm and 3000–3500 mm, but the density is a bit lower.
- **Spray (Orange)**: It also follows the bimodal trend, with peaks at the same ranges, yet it has the lowest density overall.
- The bimodal shape reveals two distinct rainfall patterns: one for low rainfall (200–500 mm) and another for high rainfall (3000–3500 mm).

**Box Plot (Right)**

- **Description**: This box plot compares the rainfall distributions for Drip, Basin, and Spray methods. The x-axis ranges from 0 to 3500 mm.
- **Analysis**:
- **Drip**: The median rainfall is about 3000 mm, with an interquartile range (IQR) from 500 to 3200 mm, and there are no notable outliers.
- **Basin**: The median is slightly lower at around 2800 mm, with an IQR from 500 to 3100 mm.
- **Spray**: This method has the lowest median at about 2500 mm, with an IQR from 500 to 3000 mm.
- The box plots reinforce the bimodal distribution, showing that most data points cluster around the high rainfall range (3000 mm), while a significant number fall into the low rainfall range (500 mm).

**Insights**

- The bimodal rainfall distribution hints at regions with varying climates, such as semi-arid areas (200–500 mm) and tropical or monsoon-heavy regions (3000–3500 mm).

- Drip irrigation tends to be linked with slightly higher average rainfall, suggesting its use in areas with more reliable monsoon conditions, possibly to enhance natural rainfall.
- The similarities in rainfall distributions across the different irrigation methods imply that the choice of method may not be heavily influenced by rainfall patterns alone; other factors like crop.

---

**Distribution of Temperature**

**Density Plot (Left)**:

**Description**

This density plot illustrates how Temperature (in °C) is distributed across Drip, Basin, and Spray irrigation methods. The x-axis spans from 0 to 250, while the y-axis indicates density.

**Analysis**

- **Drip (Purple)**: The distribution leans to the right, peaking between 25–50°C, but there are some bizarre values reaching up to 250°C.
- **Basin (Red)**: The distribution is similar, peaking around 25–50°C with outliers extending to 250°C.
- **Spray (Orange)**: The pattern here is akin to Drip and Basin, peaking around 25–50°C and featuring outliers up to 250°C.
- The occurrence of temperatures hitting 250°C points to a data quality problem, as it's rare for natural temperatures to go beyond 50°C.

**Box Plot (Right)**

**Description**

The box plot provides a comparison of Temperature distributions for Drip, Basin, and Spray. The x-axis ranges from 0 to 225.

**Analysis**

- **Drip**: The median temperature hovers around 30°C, with an interquartile range (IQR) from 25 to 40°C. Outliers reach up to 200°C.
- **Basin**: The median is similar, around 30°C, with an IQR from 25 to 40°C. Outliers also extend to 200°C.
- **Spray**: The median is a tad lower, around 28°C, with an IQR from 25 to 35°C. Outliers again reach up to 200°C.
- These box plots reinforce the data quality issue, as unrealistic outliers distort the distribution.

**Insights**

- If we consider the outliers (like 250°C) as errors, the realistic temperature range of 25–50°C suggests a tropical or subtropical climate, which fits well with crops such as Coconut.
- The similarity in temperature distributions across the different irrigation methods implies that temperature might not be a key factor in selecting an irrigation method.
- It's crucial to clean the data by removing or correcting those unrealistic temperature values (like anything above 50°C) to ensure a more accurate analysis.

---

**Distribution of Yields**

**Density Plot (Left)**

**Description**

The density plot illustrates how Yields (probably measured in kg or tons) are distributed across Drip, Basin, and Spray irrigation methods. The x-axis spans from 0 to 70,000, while the y-axis indicates density.

**Analysis**

- **Drip (Purple)**: This distribution leans to the right, peaking around 0–10,000, with another smaller peak between 50,000 and 60,000.
- **Basin (Red)**: Similar to Drip, it peaks around 0–10,000 and has a smaller peak around 50,000–60,000, but with a slightly lower density overall.
- **Spray (Orange)**: This distribution mirrors the others, peaking around 0–10,000 and again at 50,000–60,000, but it shows the lowest density of the three.
- Overall, the distribution is bimodal, indicating two distinct groups: lower yields (0–10,000) and higher yields (50,000–60,000).

**Box Plot (Right)**

**Description**

Description: The box plot provides a comparison of Yields distributions for Drip, Basin, and Spray, with the x-axis ranging from 0 to 60,000.

**Analysis**

- **Drip**: The median yield sits around 5,000, with an interquartile range (IQR) from 2,000 to 50,000. There are outliers reaching up to 60,000.
- **Basin**: The median is a bit lower at around 4,000, with an IQR from 2,000 to 45,000, and outliers also extending to 60,000.
- **Spray**: This method has the lowest median at about 3,000, with an IQR from 2,000 to 40,000, and outliers that go up to 60,000.

- The box plots reinforce the bimodal distribution, showing that while most yields are on the lower side, there are still a notable number of high-yield entries.

**Insights**

- The bimodal distribution of yields points to some variability in productivity, which could stem from differences in crop types—like Coconut versus Coffee—or even the farming practices used.
- Drip irrigation seems to lead to slightly higher average yields (with a higher median), suggesting it might be effective in boosting productivity, likely because it delivers water more efficiently.
- The fact that yield distributions are similar across different irrigation methods indicates that other factors—such as the type of crop, the soil quality, or the region—might play a bigger role in determining yields than the irrigation method itself.

---

**Distribution of Humidity**

**Density Plot (Left)**

**Description**

This density plot illustrates how humidity (in %) varies across Drip, Basin, and Spray irrigation methods. The x-axis spans from 48 to 62, while the y-axis indicates density.

**Analysis**

- **Drip (Purple)**: The distribution appears to follow a normal pattern, peaking around 55–56% and showing a slight lean towards higher humidity levels.
- **Basin (Red)**: This distribution is quite similar, also peaking around 55–56% with a slight skew.
- **Spray (Orange)**: The distribution here reflects that of Drip and Basin, peaking around 55–56% but with the lowest density overall.
- All three methods show consistent distributions, suggesting stable humidity levels.

**Box Plot (Right)**

**Description**

The box plot provides a comparison of humidity distributions for Drip, Basin, and Spray. The x-axis ranges from 50 to 58.

**Analysis**

- **Drip**: The median sits at about 55%, with an interquartile range (IQR) from 53 to 57%. There are no notable outliers.
- **Basin**: The median is also around 55%, with an IQR from 53 to 57%.
- **Spray**: Here, the median is slightly lower at around 54%, with an IQR from 52 to 56%.

- These box plots reinforce the idea of a normal distribution, with most values clustering around 55%.

**Insights**

- The stable humidity levels (hovering around 55%) across all irrigation methods hint that this dataset likely comes from a tropical or monsoon-influenced area, where humidity tends to be high.
- The absence of significant differences in humidity distributions suggests that humidity might not play a crucial role in selecting an irrigation method.
- The consistent humidity range (50–60%) is beneficial for various crops, like Coconut, which flourishes in humid environments.

---

**Distribution of Price**

**Density Plot (Left)**:

**Description**

This density plot illustrates how Price (in INR) is distributed across Drip, Basin, and Spray irrigation methods. The x-axis stretches from -5000 to 300000, while the y-axis indicates density.

**Analysis**

- **Drip (Purple)**: The distribution shows a bimodal pattern, peaking around 0–50000 and again at 200000–250000.
- **Basin (Red)**: This distribution mirrors the Drip method, with peaks at 0–50000 and 200000–250000, but it has a slightly lower density overall.
- **Spray (Orange)**: Following a similar trend, the Spray method also peaks at 0–50000 and 200000–250000, but it has the lowest density among the three.
- The bimodal shape suggests there are two distinct groups: one for lower prices (0–50000) and another for higher prices (200000–250000).

**Box Plot (Right)**

**Description**

The box plot provides a comparison of Price distributions for Drip, Basin, and Spray methods. The x-axis ranges from 0 to 250000.

**Analysis**

- **Drip**: The median price sits around 40000, with an interquartile range (IQR) from 20000 to 200000. There are outliers reaching up to 250000.
- **Basin**: The median here is a bit lower, around 35000, with an IQR from 20000 to 180000.

- **Spray**: This method has the lowest median at about 30000, with an IQR from 20000 to 150000.
- The box plots reinforce the bimodal distribution, showing that while most prices are on the lower end, there are still a notable number of high-price entries.

**Insights**

- The bimodal price distribution indicates a range of economic values, likely influenced by factors such as crop types (for instance, Coconut versus high-value crops like Cardamom) or varying market conditions.
- Drip irrigation tends to be linked with slightly higher average prices (higher median), suggesting it may be favored in more profitable farming operations or for cultivating higher-value crops.
- The similarities in price distributions across the different irrigation methods imply that the choice of irrigation method might not be the main

---

**Overall Analysis and Insights**

**Bimodal Distributions**

- When we look at Area, Rainfall, Yields, and Price, we see that they all show bimodal distributions. This means there are two distinct groups within each variable—think small versus large areas, or low versus high rainfall and yields/prices. This diversity hints at different farming operations at play, like small-scale farms compared to commercial ones, or even variations based on region.

**Data Quality Issues**

- The Temperature distribution has some unrealistic values, like 250°C, which points to a data quality problem that needs fixing. For agricultural data, we should expect temperatures to fall between 20–50°C.
- The Area distribution also has negative values, such as -1000, which are likely mistakes and should definitely be corrected.

**Irrigation Method Comparisons**

- Drip irrigation tends to show slightly higher medians for Area, Rainfall, Yields, and Price. This suggests it might be linked to larger, more productive, or more profitable farming operations.
- After we correct for outliers, Humidity and Temperature show little variation across different irrigation methods. This indicates that these environmental factors might not play a significant role in determining which irrigation method is chosen.
- The The similarities in distributions across irrigation methods for most variables imply that other factors (like crop type, soil quality, or region) could have a bigger influence on these variables than the irrigation method itself.

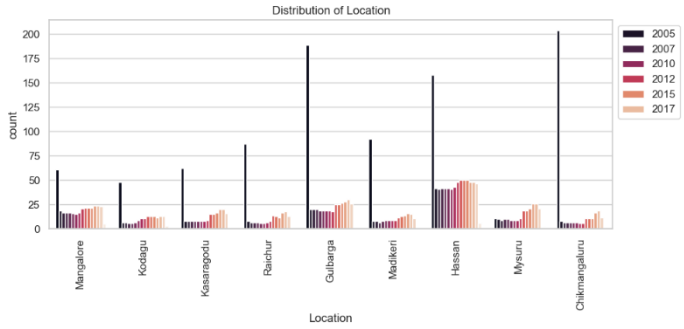
**Environmental and Economic Insights**

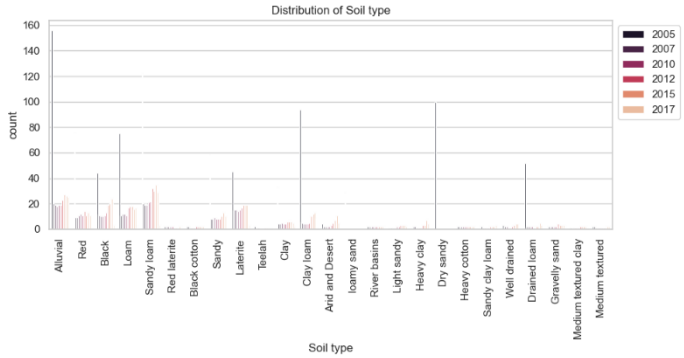
- The bimodal rainfall distribution, with ranges of 200–500 mm and 3000–3500 mm, showcases the variety of climates represented in the dataset, likely pointing to semi-arid and tropical regions.
- The stable humidity level (around 55%) and the realistic temperature range (25–50°C, after correcting outliers) suggest a tropical or subtropical climate, which aligns with a focus on crops like Coconut.
- The bimodal price distribution reveals a mix of low-value and high-value entries, which could be worth exploring further to understand the economic factors at play, such as crop type and market dynamics.

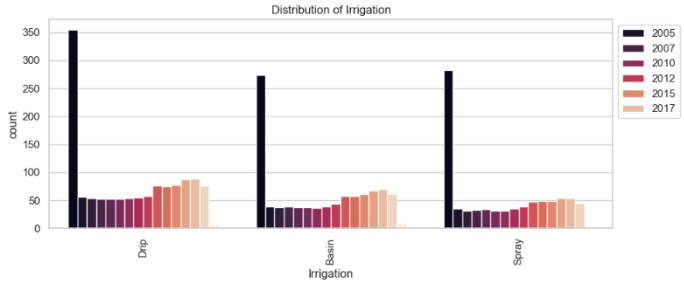
**Recommendations for Further Analysis**

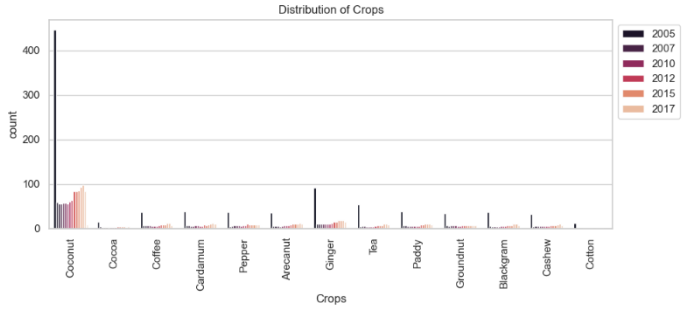
- **Data Cleaning**: It's important to tackle outliers in Temperature (like removing any values over 50°C) and Area (for instance, fixing any negative values) to make sure our analysis is spot on.
- **Cross-Variable Analysis**: Let’s dive into the connections between different variables, such as how the irrigation method affects yields or prices for certain crops (think Coconut versus Coffee).
- **Regional Analysis**: If we have location data, we should look into how rainfall, yields, and prices change by region to get a better grasp of geographical trends.
- **Sustainability Focus**: With Drip irrigation being so common, it would be great to analyze how it influences water usage, yields, and profitability, showcasing its importance in sustainable farming practices.

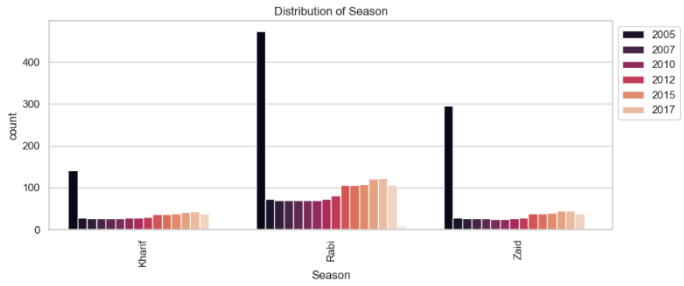
# Count plot of categorical variables vs year



**Distribution of Location**

**Description**

This stacked bar chart illustrates how often data entries appear across various locations (like Mangalore, Kodagu, Chikmagaluru, Raichur, Gulbarga, Madikeri, Hassan, Mysuru) for the years 2005, 2007, 2010, 2012, 2015, and 2017.

**Analysis**

- Dominant Locations: In 2005, Mangalore, Kodagu, and Chikmagaluru lead the pack with around 150–200 entries each. Hassan and Gulbarga also show notable counts, sitting around 100–150.

**Yearly Trends**

- 2005 (Black): This year stands out across all locations, boasting the highest counts for Mangalore, Kodagu, and Chikmagaluru.
- 2007–2017 (Purple, Red, Orange, Light Orange): The numbers for these years drop significantly, with each contributing only a small slice (usually 10–30 entries) to the total for each location.
- Hassan and Gulbarga: These areas maintain a steady presence throughout the years, with modest contributions from 2007 to 2017.
- Madikeri and Mysuru: These locations show lower overall counts, primarily from 2005, with very few entries in the following years.
- Distribution: The data is heavily weighted towards 2005, with a noticeable decline in data collection in the years that follow across all locations.

Insights

- The strong presence of 2005 hints at a significant data collection initiative that year, likely tied to a specific agricultural survey or study. This temporal bias might restrict the dataset's ability to track trends over time.
- Regions like Mangalore, Kodagu, and Chikmagaluru, known for their Coconut, Coffee, and Cardamom production, are well-represented, highlighting a focus on key agricultural areas in South India, particularly Karnataka.
- The steady yet smaller data presence in later years (2007–2017) suggests that data collection persisted but at a reduced scale, which could be worth investigating further to understand the reasons behind this decline.

**Distribution of Soil Type**

**Description**

This stacked bar chart illustrates how often different soil types—like Alluvial, Black, Red, Laterite, Sandy loam, Clay loam, and others—were recorded in the years 2005, 2007, 2010, 2012, 2015, and 2017.

**Analysis**

- Dominant Soil Types: In 2005, Sandy loam, Alluvial, and Loam stood out with the highest counts, each ranging from about 100 to 150 entries. Laterite, Black, and Red also made their mark that year, with counts between 50 and 100.
- Yearly Trends:
- 2005 (Black): This year is a standout for most soil types, showcasing the highest counts for Sandy loam, Alluvial, and Loam.
- 2007–2017 (Purple, Red, Orange, Light Orange): The numbers for these years drop significantly, with each contributing only a handful of entries (usually between 5 and 20) to the total for each soil type.
- Less Common Soil Types: Soil types such as Heavy cotton, Light sandy, Black cotton, Red clay, Red laterite, Medium textured clay, and Teelah show very low counts, primarily from 2005, with little to no contributions in the following years.
- Distribution: The data is heavily weighted towards 2005, with a noticeable decline in data collection in the years that followed for all soil types.

**Insights**

- The strong presence of Sandy loam, Alluvial, and Loam in 2005 likely reflects a focus on crops like Coconut, which thrive in these well-drained, fertile soils.
- The variety of soil types, even those with fewer entries, suggests that the dataset captures a wide range of agricultural settings, from fertile plains (Alluvial) to more challenging terrains (Laterite, often found in the Western Ghats).
- The heavy emphasis on 2005 indicates that most soil type data was gathered that year, which might limit our understanding of how soil conditions have changed over time.

---

**Distribution of Irrigation**

**Description**

This stacked bar chart illustrates how often different irrigation methods—Drip, Basin, and Spray—were used in the years 2005, 2007, 2010, 2012, 2015, and 2017.

**Analysis**

- Dominant Irrigation Method: In 2005, Drip irrigation took the lead with about 300–350 entries, followed by Basin with around 250, and Spray with roughly 200.

**Yearly Trends**

- 2005 (Black): This year stands out for all irrigation methods, showcasing the highest counts for Drip, Basin, and Spray.
- 2007–2017 (Purple, Red, Orange, Light Orange): The numbers for these years drop significantly, with each contributing only a small fraction (usually 20–50 entries) to the total for each method.
- Consistency Across Methods: All three irrigation methods follow a similar trend, peaking in 2005 and then experiencing a sharp decline in the following years.
- Distribution: The data is heavily concentrated in 2005, with a noticeable decrease in data collection in the years that followed for all irrigation methods.

**Insights:**

- The strong presence of Drip irrigation in 2005 highlights a shift towards water-efficient farming practices, which are essential in areas facing inconsistent rainfall or water shortages.
- The notable use of Basin and Spray irrigation suggests that both traditional and alternative methods remain common, possibly reflecting a variety of agricultural practices or regional preferences.
- The focus on 2005 limits our ability to track changes in irrigation practices over time, but the data from that year offers a valuable glimpse into irrigation choices during that period.

---

4**. Distribution of Crops**

**Description**

This stacked bar chart illustrates how often various crops—like Coconut, Ginger, Coffee, Cardamom, Arecanut, Tea, Pepper, Paddy, Blackgram, Groundnut, Cashew, Cocoa, and Cotton—were recorded in the years 2005, 2007, 2010, 2012, 2015, and 2017.

**Analysis**

- Dominant Crop: In 2005, Coconut stands out with around 400 entries, far surpassing the other crops.
- Secondary Crops: Ginger, Coffee, Cardamom, and Arecanut also show significant numbers in 2005, each ranging from 50 to 100 entries. In contrast, crops like Tea, Pepper, Paddy, Blackgram, Groundnut, Cashew, Cocoa, and Cotton have much lower counts, usually under 50.

**Yearly Trends**

- 2005 (Black): This year is a standout, showcasing the highest counts for Coconut, Ginger, Coffee, and Cardamom.

- 2007–2017 (Purple, Red, Orange, Light Orange): The counts for these years drop significantly, with each contributing only a small number of entries (typically between 5 and 20) for each crop.
- Less Common Crops: Crops such as Cotton, Cocoa, and Cashew show very low counts, primarily from 2005, with little to no data from the following years.
- Distribution: The data is heavily weighted towards 2005, with a noticeable decline in data collection for the subsequent years across all crops.

**Insights**

- Coconut's dominance in 2005 highlights the dataset's emphasis on Coconut farming, likely reflecting the agricultural practices in regions like Karnataka and Kerala.
- The inclusion of crops like Ginger, Coffee, Cardamom, and Arecanut points to a variety of agricultural practices, possibly from hilly areas such as Kodagu and Chikmagaluru where these crops thrive.
- The focus on 2005 limits our ability to track crop trends over time, but the data from that year offers a valuable glimpse into crop preferences during that period.

---

**Distribution of Season**

**Description**

This stacked bar chart illustrates how often agricultural seasons (Kharif, Rabi, Zaid) occurred in the years 2005, 2007, 2010, 2012, 2015, and 2017.

**Analysis**

- Dominant Season: In 2005, Rabi stands out with about 400 entries, followed by Zaid with around 200, and Kharif trailing behind at about 100.

**Yearly Trends**

- 2005 (Black): This year is a clear leader across all seasons, boasting the highest counts for Rabi, Zaid, and Kharif.
- 2007–2017 (Purple, Red, Orange, Light Orange): The numbers for these years drop significantly, with each contributing only a small fraction (usually 20–50 entries) to the total for each season.
- Consistency Across Seasons: All three seasons follow a similar trend, peaking in 2005 and then sharply declining in the following years.
- Distribution: The data is heavily weighted towards 2005, showing a notable decrease in data collection in the years that follow for all seasons.

**Insights**

- Rabi's prominence in 2005 indicates that the dataset likely focuses on crops or areas where the Rabi season (winter, October to March) plays a crucial role. This is intriguing,

especially since Coconut is typically linked to the Kharif season (monsoon, June to September).

- The strong emphasis on 2005 limits our ability to track seasonal trends over time, but the data from that year does give us a glimpse into seasonal preferences.
- The focus on Rabi might highlight particular agricultural practices, like Coconut farming with irrigation during the drier Rabi season, or it could suggest a bias in the dataset towards Rabi crops.

---

**Overall Analysis and Insights**

**Temporal Bias**

- The dataset shows a significant lean towards the year 2005 across all variables, including Location, Soil Type, Irrigation, Crops, and Season. There are noticeably fewer entries from the years 2007 to 2017. This likely points to a major data collection effort that took place in 2005, possibly linked to a specific agricultural survey or study.
- The lack of data from later years restricts the dataset's ability to track trends over time, which could pose challenges for any longitudinal analyses.

**Agricultural Focus**

- There's a clear emphasis on Coconut cultivation, as highlighted by its prominence in the "Distribution of Crops" chart. This aligns perfectly with regions like Mangalore, Kodagu, and Chikmagaluru, which are well-known for their Coconut production.
- The prevalence of Sandy loam, Alluvial, and Loam soils further supports this focus, as these soil types are ideal for growing Coconut.

**Seasonal and Irrigation Patterns**

- The Rabi season stands out, especially considering the focus on Coconut, which usually flourishes in monsoon conditions. This might suggest some unique farming practices, like Rabi-season Coconut farming with irrigation, or it could indicate a bias in the dataset towards Rabi crops.
- The preference for Drip irrigation points to a commitment to water-efficient farming, which is crucial in areas facing variable rainfall or water shortages.

**Data Representation and Gaps**

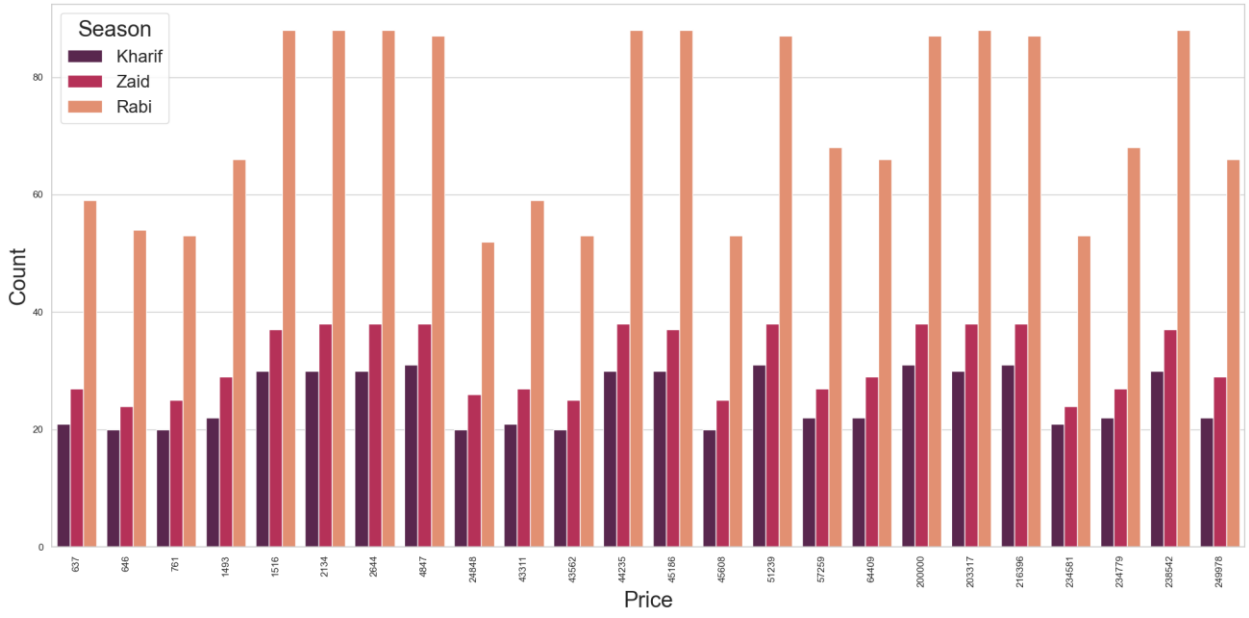
- The dataset is unevenly spread across the years, with 2005 taking the lead. Other categories, such as less common crops like Cotton and Cocoa, certain soil types like Teelah, and seasons like Kharif, are not well represented in the later years.
- This uneven distribution could limit the dataset's usefulness for studying trends over time or for exploring less common categories. Future data collection efforts should strive for a more balanced representation across the years.

**Recommendations for Further Analysis**

- Focus on 2005 Data: Since 2005 is a key year, let's take a closer look at it to really grasp the agricultural practices, environmental conditions, and economic outcomes from that time.
- Cross-Variable Analysis: Let's dig into the connections between different factors, like how soil type influences crop selection (for instance, Coconut on Sandy loam versus Coffee on Laterite) or how different irrigation methods affect productivity across various seasons.
- Regional Insights: It's important to examine how soil types, crops, seasons, and irrigation techniques differ by region (like Coconut in Mangalore compared to Coffee in Kodagu) to uncover geographical trends.
- Sustainability Focus: With Drip irrigation being so common, let's analyze how it impacts water usage and crop yield, particularly for Coconut, to showcase sustainable farming practices.

**Count plot of numerical variables vs season**

**Price Distribution by Season**




**Description**

A stacked histogram that illustrates the distribution of prices (in INR) across the Kharif, Rabi, and Zaid seasons. Prices range from 187 to 29,289 INR.

**Analysis**

- The majority of prices fall below 5,000 INR, with Rabi (represented in orange) leading the way in all categories, making up most of the counts (for instance, 60–80 in the lower price ranges).

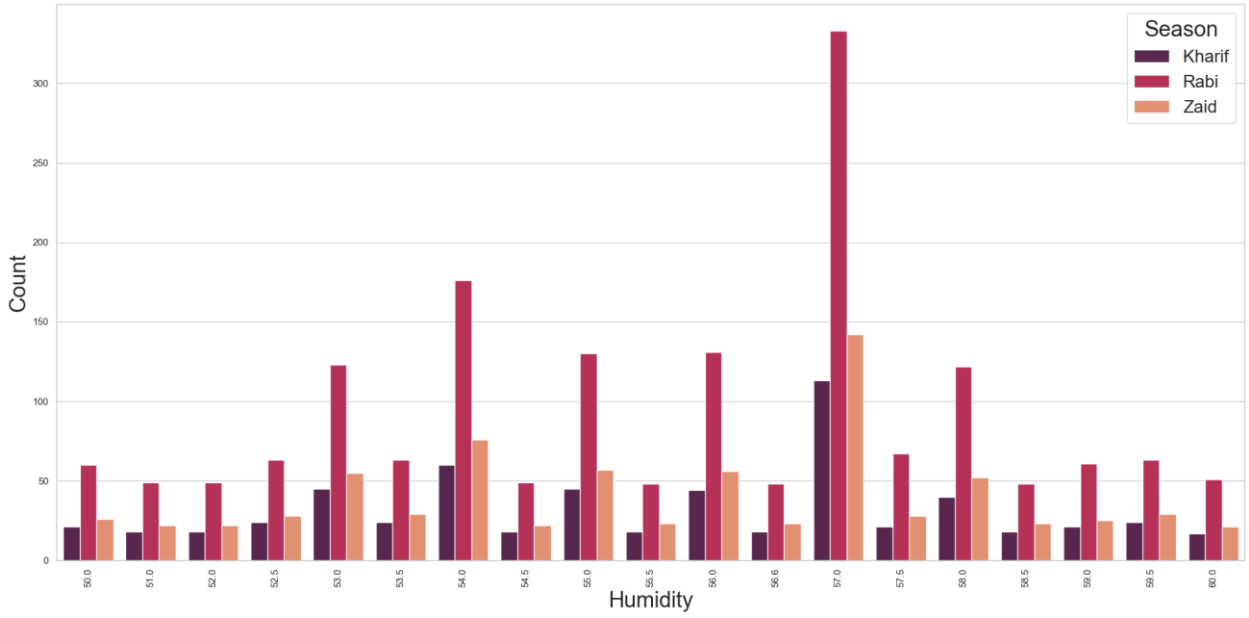
- Kharif (shown in purple) and Zaid (in red) contribute less overall, but Zaid shows up a bit more often in the higher price ranges (like 20,000–29,289 INR, where it hits counts of 10–20).
- The distribution leans to the right, featuring a long tail of higher prices, which suggests there are a few high-value entries, likely for premium crops such as Cardamom or Coffee.

**Insights**

- Rabi's strong presence indicates that this dataset might be centered around crops or areas where the Rabi season yields better economic returns.
- The occurrence of high prices in Zaid could point to specific high-value summer crops or unique market conditions during that time.

---

**Humidity Distribution by Season**




**Description**

This stacked histogram illustrates the distribution of Humidity (%) across the Kharif, Rabi, and Zaid seasons, with humidity levels ranging from 50% to 91%.

**Analysis**

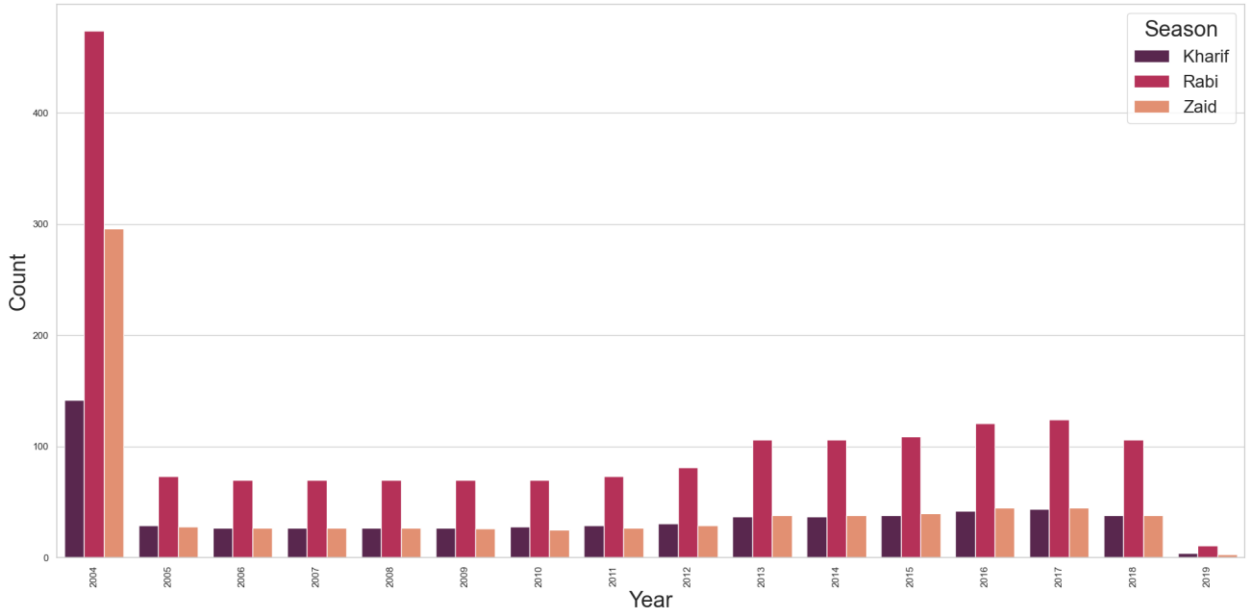
- Humidity tends to peak around 55–56%, with the Rabi season (represented in orange) taking the lead, showing counts between 150 and 200 in the peak bins.
- Kharif (in purple) and Zaid (in red) contribute less, but Zaid shows a slight uptick at the higher humidity levels (like 90–91%), reaching counts of 20–30.
- The overall distribution resembles a normal curve, centered around 55%, but with a slight lean towards the higher humidity levels.

**Insights**

- The consistent humidity range of 50–60% across the seasons fits well with a tropical climate, which is ideal for crops such as Coconut, likely the most prevalent in this dataset.
- Zaid's minor rise at elevated humidity levels might indicate summer conditions in certain areas where moisture is more abundant.

---

**Year Distribution by Season**




**Description**

A stacked histogram that illustrates the distribution of years across the Kharif, Rabi, and Zaid seasons, covering the years from 2004 to 2019.

**Analysis**

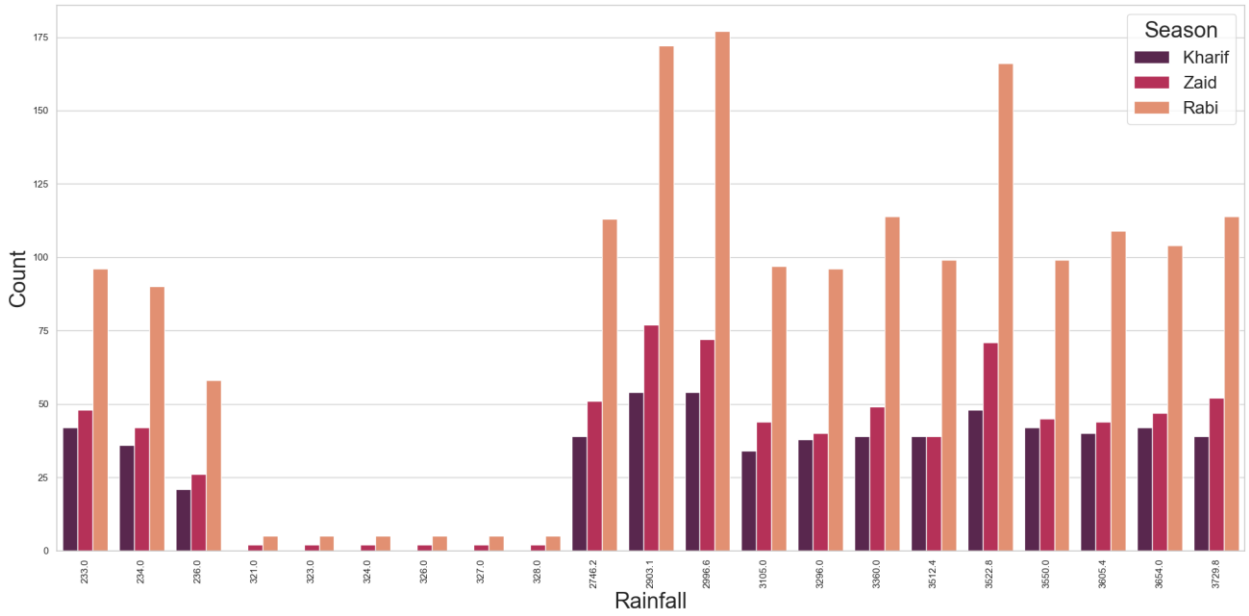
- In 2004, we see the highest count, around 450, with Rabi (in orange) leading the pack at about 300, followed by Zaid (in red) at roughly 100, and Kharif (in purple) trailing behind at around 50.
- After 2004, the counts take a noticeable dip, with the years from 2005 to 2019 showing only small contributions, typically between 20 and 50 per year, mostly from Rabi.
- Kharif and Zaid are less represented throughout the years, with very few entries in the later years—2019, for instance, has almost no data.

**Insights**

- The strong emphasis on 2004 indicates a significant data collection effort that year, likely stemming from a one-time agricultural survey, which makes it hard to analyze trends over time.

- Rabi's consistent presence across the years highlights the dataset's focus on winter season farming.

**Rainfall Distribution by Season**




**Description**

A stacked histogram that illustrates the distribution of rainfall (in mm) across the Kharif, Rabi, and Zaid seasons, with rainfall amounts ranging from 230 to 3658 mm.

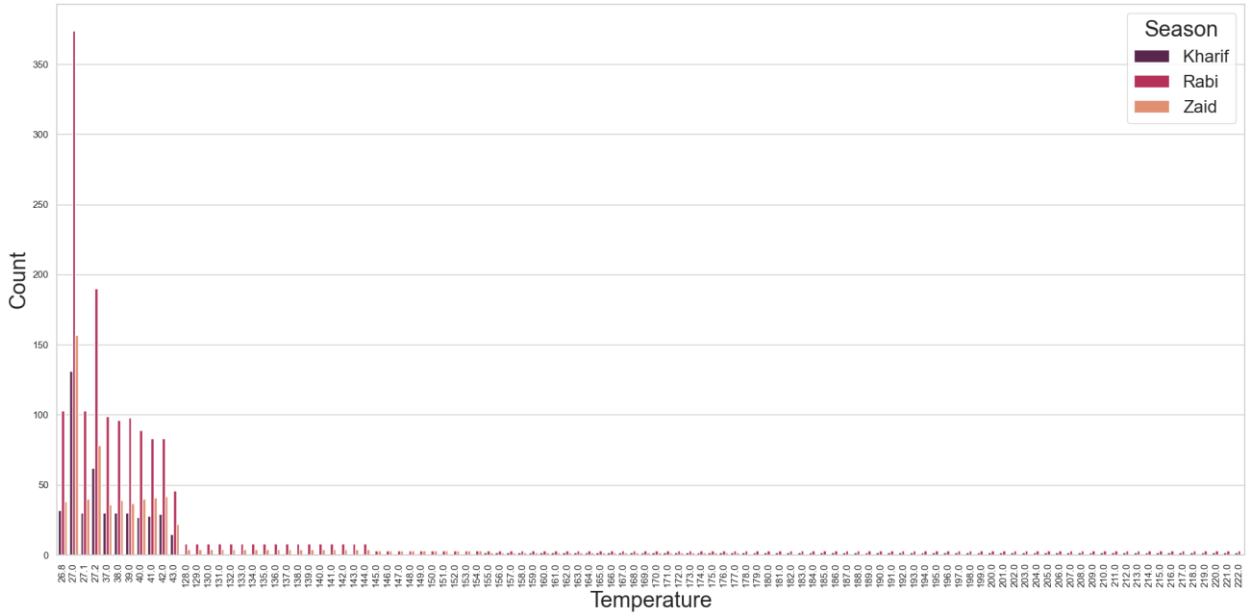
**Analysis**

- A stacked histogram that illustrates the distribution of rainfall (in mm) across the Kharif, Rabi, and Zaid seasons, with rainfall amounts ranging from 230 to 3658 mm.
- The Rabi season (marked in orange) stands out in both peaks, contributing around 50 in the lower range and about 125 in the higher range, followed by Zaid (in red) and Kharif (in purple).
- Kharif shows the least contribution, particularly in the higher rainfall categories (for instance, in the 3000–3658 mm range, Kharif counts drop below 20).

**Insights**

- This bimodal rainfall pattern highlights the variety of climates: semi-arid (230–500 mm) and tropical/monsoon-heavy (3000–3658 mm), which aligns well with regions in South India.
- It's quite surprising to see Kharif's low representation in the high rainfall categories, especially since it's the monsoon season. This might indicate a bias in the dataset towards Rabi-season data.

**Temperature Distribution by Season**




**Description**

A stacked histogram that illustrates the distribution of temperatures (°C) across the Kharif, Rabi, and Zaid seasons. The temperature spans from 27 to 222°C.

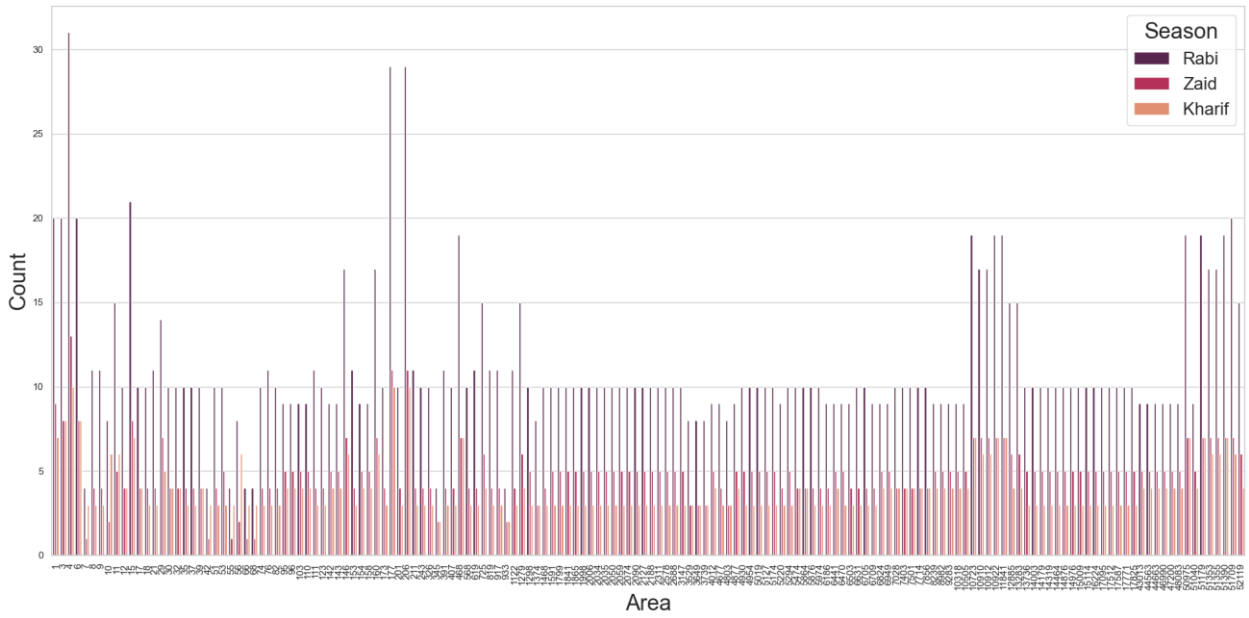
**Analysis**

- The majority of temperatures fall between 27–50°C, with Rabi (in orange) taking the lead, showing counts between 100–150 in this range.
- Some values, like 222°C, are clearly unrealistic and point to data errors; the expected range for agricultural purposes should be around 20–50°C.
- Kharif (in purple) and Zaid (in red) contribute less, with counts between 20–50 in the realistic range.

**Insights**

- The realistic temperature range of 27–50°C indicates a tropical or subtropical climate, which is ideal for growing crops such as coconut.
- To ensure accurate analysis, it's essential to clean the data by removing outliers like the 222°C reading.

**Area Distribution by Season**




**Description**

A stacked histogram that illustrates the distribution of area across the Kharif, Rabi, and Zaid seasons. The area ranges from -1000 to 6000, which is likely measured in hectares.

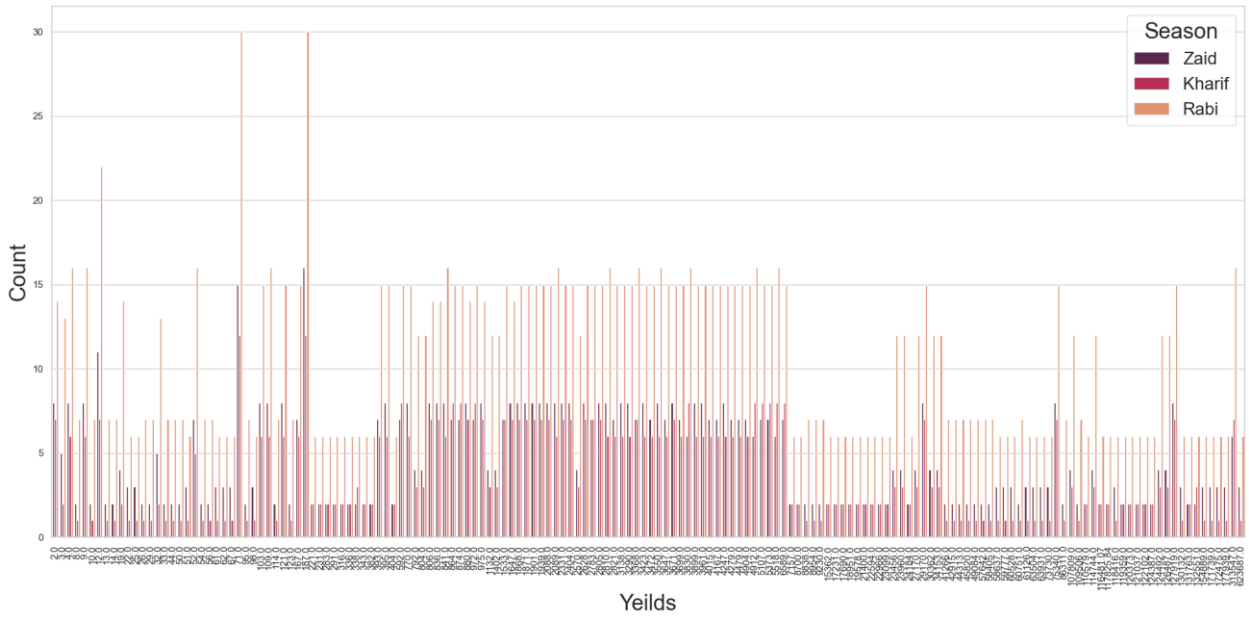
**Analysis**

- We see a bimodal distribution with peaks between 0–1000 (with counts around 15–20) and 4000–6000 (counts of about 10–15).
- Rabi (in orange) takes the lead at both peaks, contributing roughly 10–15 in each, followed by Zaid (in red) and Kharif (in purple).
- Negative values, like -1000, point to data errors; the realistic range should be between 0 and 6000.

**Insights**

- The bimodal area distribution indicates a blend of small-scale and large-scale farms, with Rabi being the most prominent.
- We need to clean up the data to fix those negative area values.

**Yields Distribution by Season**




**Description**

A stacked histogram that illustrates the distribution of yields across the Kharif, Rabi, and Zaid seasons. The yields range from 0 to 70,000 (likely in kg or tons).

**Analysis**

- We see a bimodal distribution with two peaks: one between 0–10,000 (with counts around 10–15) and another between 50,000–70,000 (with counts around 5–10).
- The Rabi season (marked in orange) stands out in both peaks, contributing about 10 in the lower range and around 5 in the higher range, followed by Zaid (in red) and Kharif (in purple).
- Kharif contributes the least, particularly in the higher yield categories (like 50,000–70,000, where Kharif counts drop below 2).

**Insights**

- The bimodal yield distribution shows that there's some variability in productivity, which could be linked to different crop types, like Coconut versus Coffee.
- Rabi's prominence hints at greater productivity during the winter months, likely thanks to irrigation or more favorable growing conditions.

---

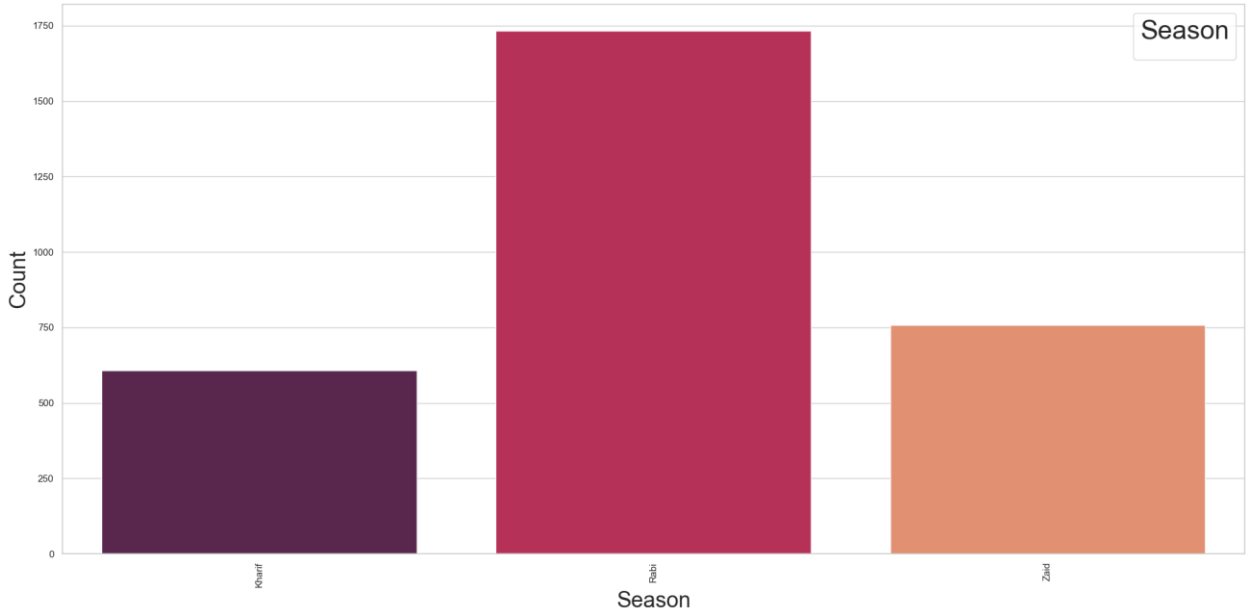
**Insights for Report**

- Rabi Dominance: The Rabi season consistently takes the lead when it comes to Price, Humidity, Year, Rainfall, Temperature, Area, and Yields. This trend suggests a bias in the dataset towards winter farming, likely influenced by irrigation methods or a focus on specific crops, like Coconut, which thrive in Rabi with drip irrigation.

- Data Quality Issues: There are some errors in the data, such as a temperature reading of 222°C and negative values for Area. These inaccuracies need to be addressed to ensure a reliable analysis.
- Bimodal Patterns: The distributions of Rainfall, Area, and Yields are bimodal, highlighting the variety in environmental conditions—think semi-arid versus tropical—as well as differences in farming scales, from small to large operations.
- Underrepresentation: The Kharif and Zaid seasons are not well represented in the data, which limits our ability to analyze trends during the monsoon and summer months. This is particularly important for crops like Coconut that depend on monsoon rainfall.
- Temporal Bias: The data is heavily skewed towards 2004, indicating that it may have been collected in a single effort. This limits our ability to conduct a thorough longitudinal analysis, so we should focus on the insights from the 2004 data.

**Countplot categorical variables vs season**

**Season Distribution**




**Description**

A histogram that illustrates the number of entries for the Kharif, Rabi, and Zaid seasons.

**Analysis**

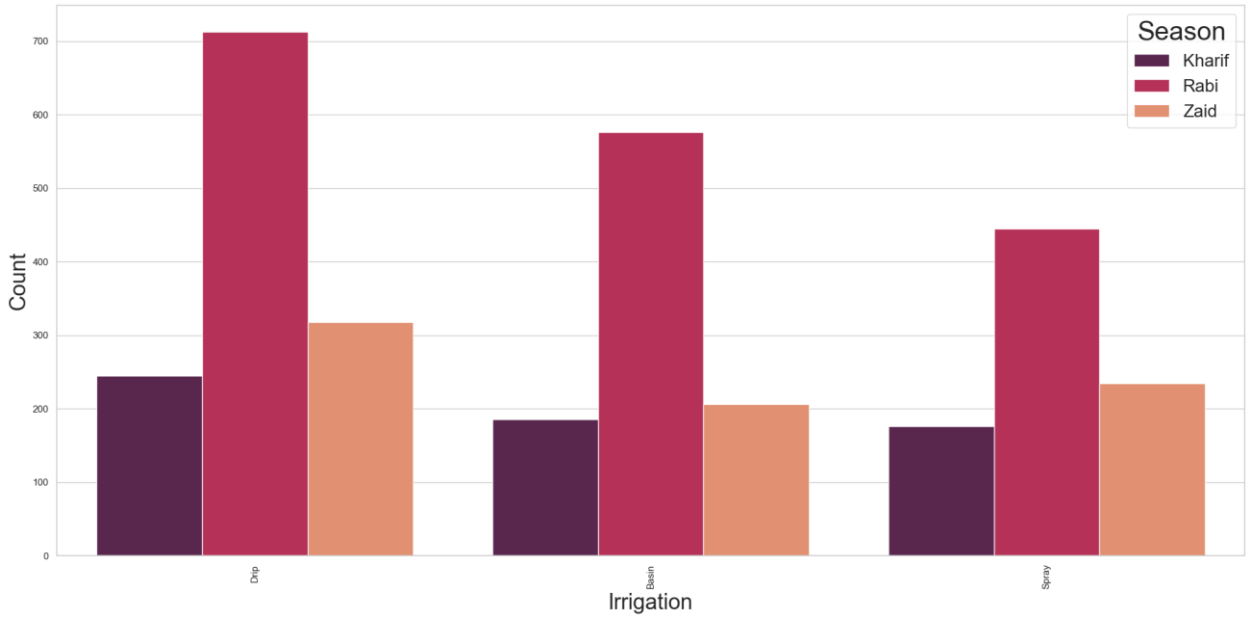
- Rabi (in red) leads the pack with around 1500 entries.
- Zaid (in orange) comes in with about 750 entries.
- Kharif (in purple) trails behind with roughly 500 entries.

**Insights**

- Rabi stands out, accounting for about 60% of the dataset, which highlights a significant emphasis on winter season farming.
- The low representation of Kharif (20%) is quite unexpected for a monsoon season, hinting at either a sampling bias or a preference for crops that don't heavily depend on monsoon rains.
- Zaid's moderate share (30%) suggests there's some activity during the summer season, likely for certain crops or in specific regions.

---

**Irrigation Distribution by Season**




**Description**

A stacked histogram that displays the counts for Drip, Basin, and Spray irrigation methods across the Kharif, Rabi, and Zaid seasons.

**Analysis**

- Drip irrigation stands out with the highest count, around 700, where Rabi (in red) contributes about 500, Zaid (in orange) adds roughly 100, and Kharif (in purple) also comes in at around 100.
- Basin irrigation follows with about 600 entries, with Rabi at around 400, Zaid at about 100, and Kharif again at around 100.
- Spray irrigation has around 500 entries, with Rabi at approximately 350, Zaid at about 100, and Kharif at just 50.

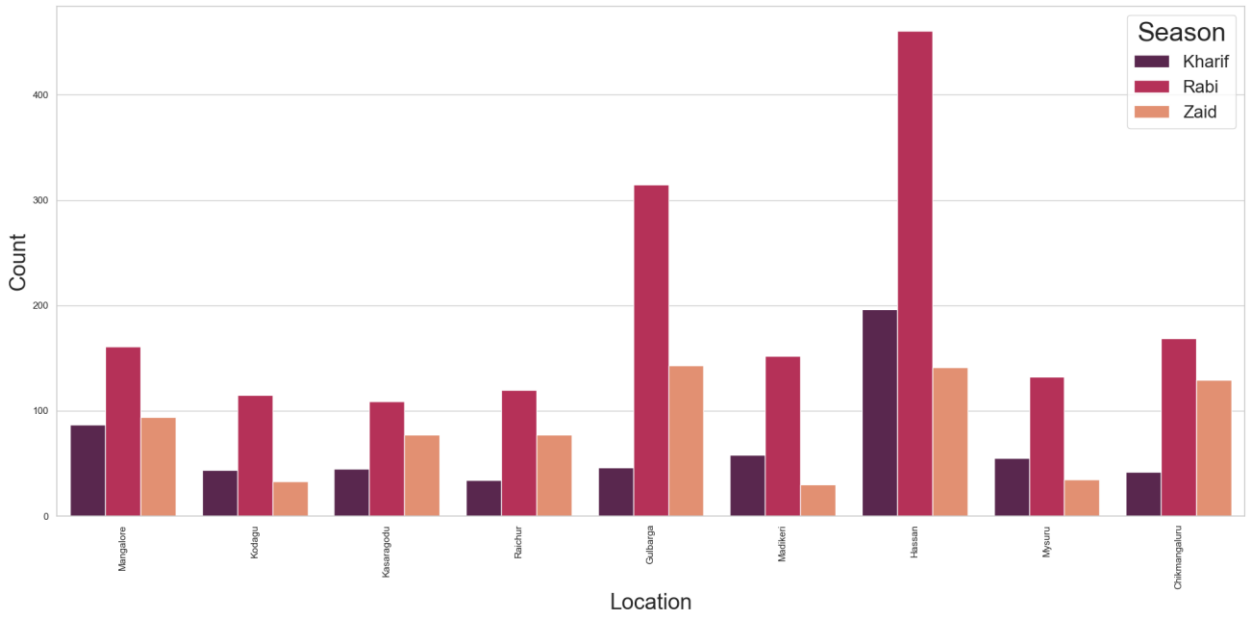
**Insights**

- Drip irrigation is clearly the most popular method, likely because it's so efficient with water—something that's really important in areas where rainfall can be unpredictable.

- The strong presence of Rabi across all methods indicates that irrigation is heavily relied upon during the winter months, possibly to help crops like Coconut thrive when it's drier.
- The lower counts for Kharif make sense given its timing with the monsoon, where natural rainfall likely lessens the need for additional irrigation.

---

**Location Distribution by Season**




**Description**

A stacked histogram that displays counts from various locations (Mangalore, Kodagu, Kesargodu, Raichur, Gulbarga, Madikeri, Hassan, Mysuru, Chikmagaluru) for the Kharif, Rabi, and Zaid seasons.

**Analysis**

- Chikmagaluru stands out with the highest count at around 450, where Rabi (in red) accounts for about 300, Zaid (in orange) is around 100, and Kharif (in purple) is roughly 50.
- Mangalore and Kodagu are next, each with about 200, where Rabi is around 150, Zaid is between 30 and 40, and Kharif is about 20 to 30.
- Other areas like Raichur, Gulbarga, and Madikeri show counts ranging from 50 to 150, with Rabi leading the way (around 40 to 100), Zaid at about 20, and Kharif between 10 and 15.

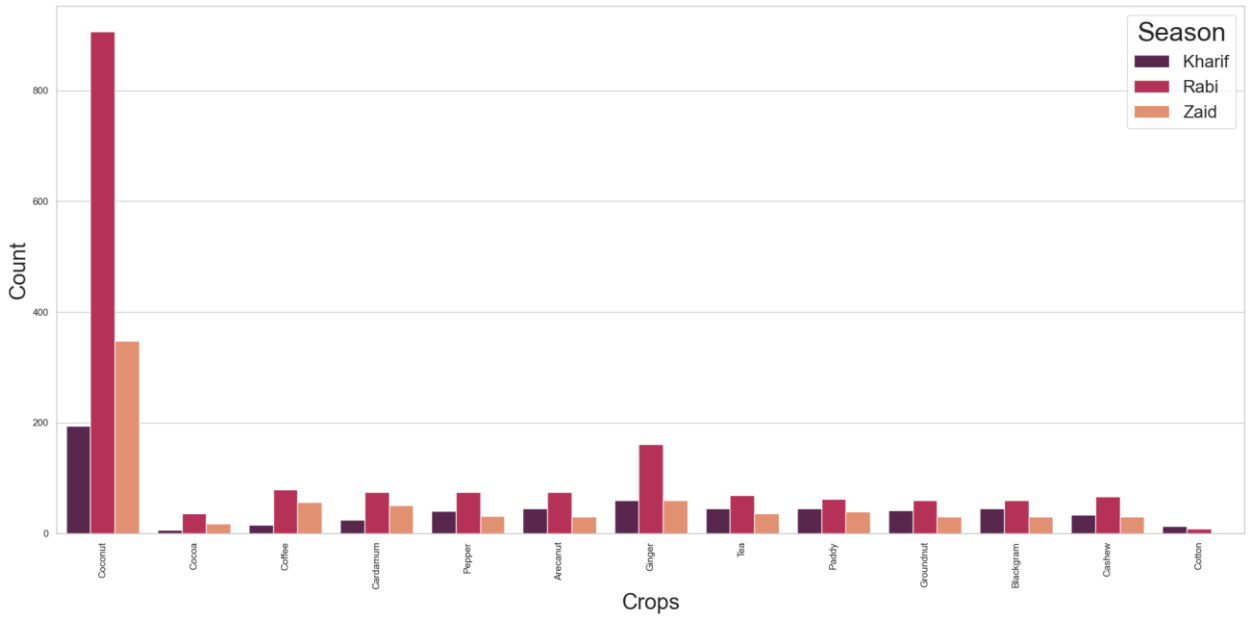
**Insights**

- The strong presence of Rabi across all locations highlights the dataset's emphasis on winter farming, likely influenced by irrigation practices in these areas.

- Chikmagaluru, Mangalore, and Kodagu, which are famous for their Coconut, Coffee, and Cardamom production in Karnataka, are well-represented, fitting perfectly with the agricultural theme of the dataset.
- The low numbers for Kharif across these locations indicate a possible under-representation of monsoon-season farming, which is quite surprising for these regions.

---

**Crops Distribution by Season**




**Description**

A stacked histogram that displays the counts for various crops, including Coconut, Cocoa, Coffee, Cardamom, Pepper, Arecanut, Ginger, Tea, Paddy, Groundnut, Blackgram, Cashew, and Cotton, across the Kharif, Rabi, and Zaid seasons.

**Analysis**

- Coconut leads the pack with the highest count, around 900, with Rabi (marked in red) at about 600, Zaid (in orange) at roughly 200, and Kharif (in purple) at around 100.
- Ginger, Coffee, and Cardamom each have counts ranging from 100 to 150, with Rabi showing about 80 to 100, Zaid around 20 to 30, and Kharif between 10 to 20.
- Other crops, like Paddy, Groundnut, and Blackgram, have counts below 50, primarily in Rabi (about 20 to 30), while Kharif and Zaid are below 10.

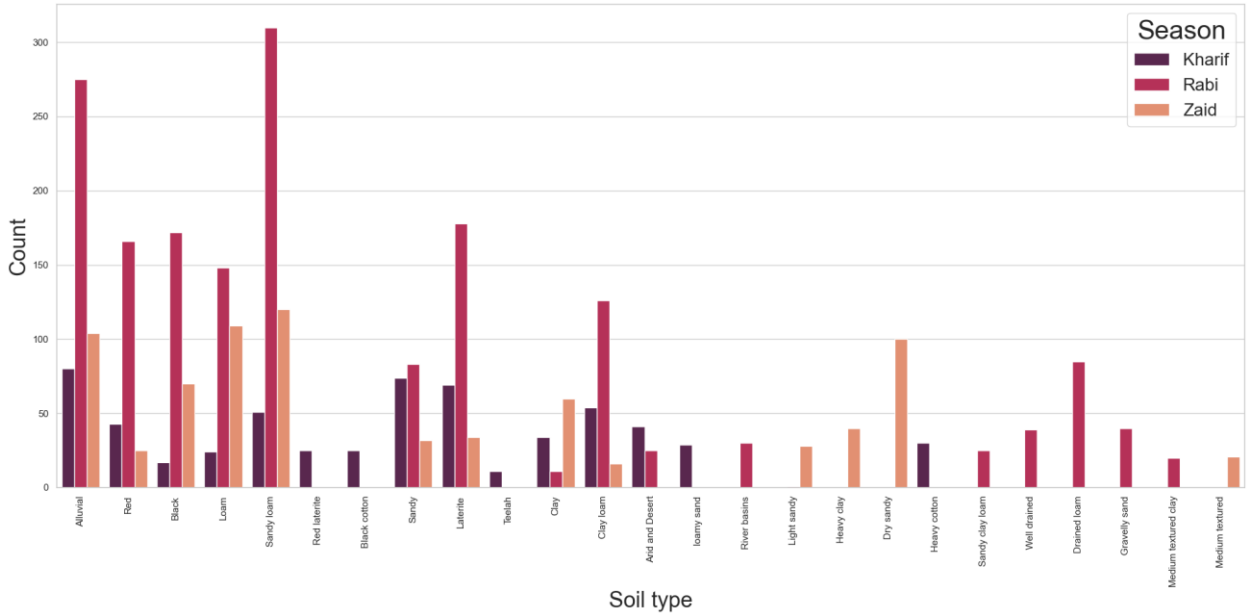
**Insights**

- Coconut's strong presence highlights the dataset's emphasis on this crop, likely mirroring the agricultural practices found in South Indian states such as Karnataka.
- The high numbers for Coconut during the Rabi season suggest that irrigation is being utilized to cultivate this crop in winter, which is unusual since it typically thrives in the monsoon.

- The low counts for the other crops point to a limited focus, which restricts our understanding of crop diversity throughout the different seasons.

---

**Soil Type Distribution by Season**




**Description**

A stacked histogram that displays the counts of various soil types (like Alluvial, Red, Black, Loam, Sandy loam, Laterite, Black cotton, Sandy, Laterite, Teelah, Clay loam, Desert sands, River loam, Light sandy, Heavy cotton, Dry well-drained clay loam, Sandy loam, Gravelly loam, and Medium textured clay) across the Kharif, Rabi, and Zaid seasons.

**Analysis**

- Sandy loam takes the lead with the highest count at around 300, followed by Rabi (in red) at about 200, Zaid (in orange) at roughly 50, and Kharif (in purple) at around 30.
- Alluvial and Loam are next, each with counts near 250, where Rabi sits at about 150–200, Zaid at around 30–40, and Kharif at about 20–30.
- Laterite, Black, and Red soils show counts between 100 and 150, with Rabi around 80–100, Zaid at about 20, and Kharif at around 10–15.
- Some less common soil types (like Teelah and Medium textured clay) have counts below 50, primarily in Rabi (around 20–30), while Kharif and Zaid are both below 10.

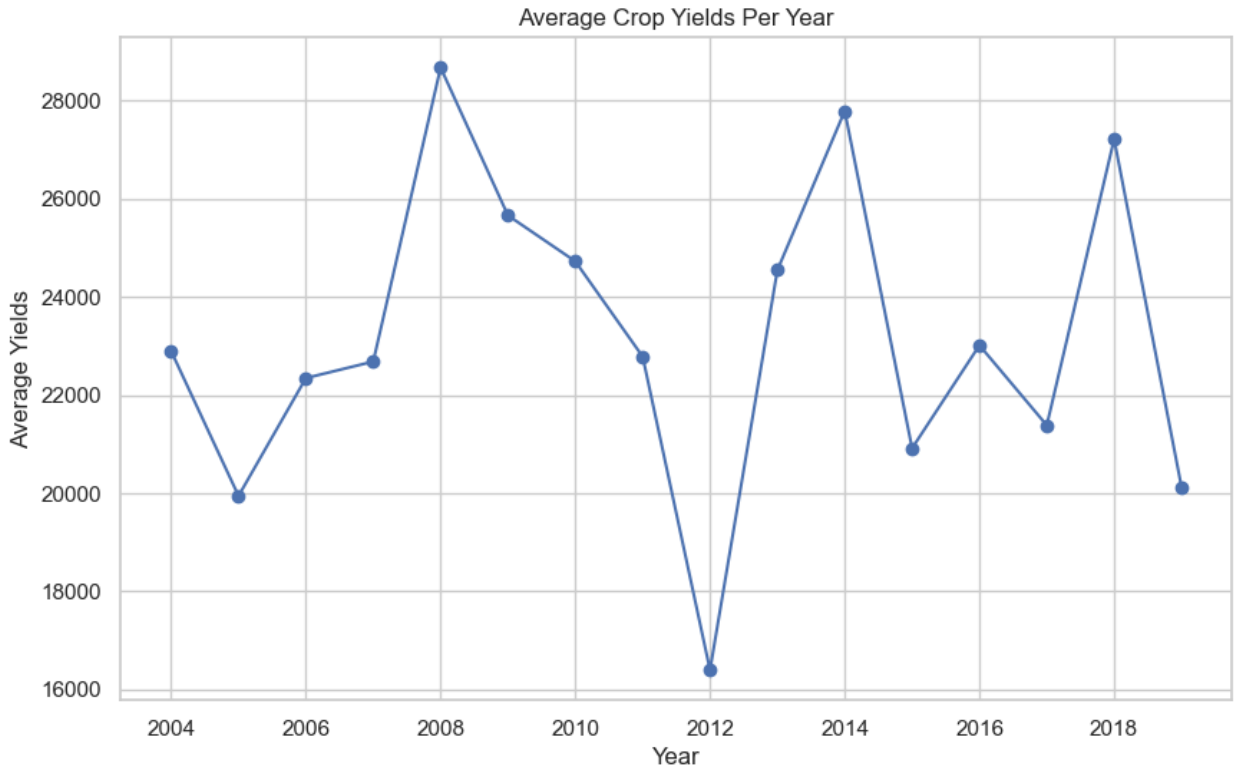
**Insights**

- Rabi stands out across all soil types, making up 60–70% of the counts, which highlights the dataset's focus on the winter season.
- Sandy loam, Alluvial, and Loam are the most common, making them ideal for Coconut cultivation, which aligns with the dataset's crop focus.

- The low representation of Kharif in soils suited for the monsoon (like Clay loam and River loam) hints at a possible sampling bias, as these soils should be more prominent during the monsoon season.

**Other visualizations**

**Average crop yields per year**




The graph presents a line plot that illustrates the average crop yields (on the y-axis) from 2004 to 2018 (on the x-axis). The y-axis spans from 16,000 to 28,000 units, which likely indicates crop yield in kilograms or tons, although the specific unit isn't mentioned. Let's break down the trends in detail:

- **2004**: Yields kick off at about 24,000 units.
- **2005**: There's a sharp drop to roughly 20,000 units.
- **2006–2008**: Yields bounce back, hitting a peak of around 28,000 units in 2008.
- **2009–2010**: A significant decline occurs, reaching a low of about 16,000 units in 2010.
- **2011–2013**: Yields climb again, peaking at approximately 28,000 units in 2013.
- **2014–2016**: Yields show some fluctuation, dipping to around 22,000 units in 2014, rising to 26,000 units in 2015, and then falling back to 22,000 units in 2016.
- **2017–2018**: Yields surge to 28,000 units in 2017, only to drop sharply to about 20,000 units in 2018.

This graph clearly illustrates the significant ups and downs in average crop yields over the 15-year span, with notable highs in 2008, 2013, and 2017, and lows in 2005, 2010, and 2018.

**Analysis of the Graph**

**1. Trends in Crop Yields**

The graph shows quite a bit of ups and downs in average crop yields over the years, hinting that various external factors likely have a big impact on agricultural productivity. Let's dive into the main trends:

**High Variability**: The yields swing dramatically, suggesting that elements like weather, farming practices, or economic situations could be affecting productivity. The difference between the highest (28,000 units) and lowest (16,000 units) yields is quite significant, highlighting the inconsistent performance from year to year.

- **Peaks in 2008, 2013, and 2017**: These years stand out with the highest average yields, reaching around 28,000 units. These peaks might be due to favorable conditions, like just the right weather (think adequate rainfall and moderate temperatures), better farming techniques, or improved access to resources such as fertilizers or irrigation.
- **Troughs in 2005, 2010, and 2018**: The lowest yields are seen in these years, with 2010 experiencing the biggest drop at 16,000 units. These declines could be tied to tough conditions, like droughts, too much rain, extreme temperatures, pest issues, or economic hurdles that made it hard for farmers to invest in necessary inputs.

**2. Potential Influencing Factors**

Although the graph doesn't give us specific details about environmental or agricultural factors, we can make some educated guesses based on what we know about farming:

- **Weather Conditions**: The noticeable drops in 2005, 2010, and 2018 might be linked to some tough weather events. Droughts or floods can really hurt crop yields since many plants need just the right amount of rainfall and temperature to grow well. On the flip side, the highs in 2008, 2013, and 2017 could suggest those were years with perfect weather for growing, thanks to balanced rainfall and temperatures.
- **Agricultural Practices**: Advances in farming methods, like using efficient irrigation systems (think drip irrigation), better soil management, or planting high-yield crop varieties, could help explain those yield peaks. Conversely, the lower points might show times when these practices weren't as common or when farmers struggled to get the resources they needed.
- **Economic and Policy Factors**: Changes in yields might also be affected by economic situations, such as shifts in the prices of farming inputs (like seeds and fertilizers) or government policies (like subsidies or support for irrigation projects). For example, the dip in 2010 could line up with a tough economic period or a decrease in agricultural support, while the peaks might correspond with years when there was more investment in farming.

**3. Year-to-Year Patterns**

The graph reveals a cyclical trend, showcasing recovery phases that follow periods of decline:

- **2004–2010**: After a dip in 2005, yields bounce back and hit their highest point in 2008, only to take a nosedive by 2010. This hints at a time of instability, likely caused by unpredictable weather or agricultural hurdles.
- **2011–2016**: Following the downturn in 2010, yields start to recover, peaking again in 2013, but then they fluctuate between 22,000 and 26,000 units. This era reflects moderate stability with some ups and downs, suggesting a blend of good and tough conditions.
- **2017–2018**: The dramatic spike in 2017, followed by a sharp decline in 2018, is particularly noteworthy. The peak in 2017 could indicate a year of outstanding conditions or the widespread use of better practices, while the drop in 2018 might point to an unexpected setback, like a natural disaster or an economic shock.

**4. Long-Term Trends**

When we look at the data over the past 15 years, it's clear there isn't a straightforward upward or downward trend in average yields. Instead, we see a pattern of fluctuations that hover around a central range of about 20,000 to 28,000 units. This inconsistency hints that while agricultural practices may have improved over time, these advancements are frequently balanced out by external challenges like climate changes or economic issues.

---

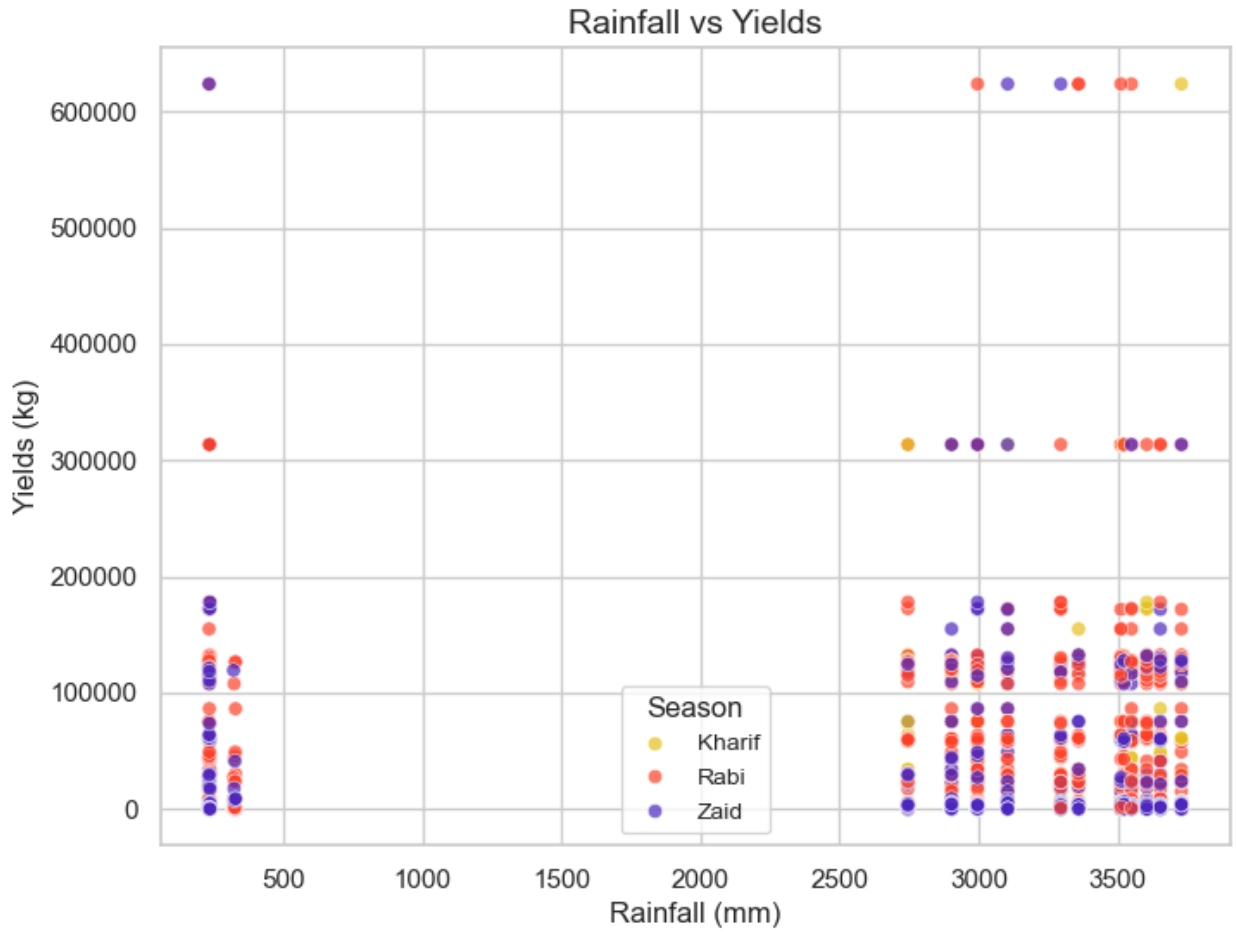
**Key Insights**

- **Significant Volatility in Yields**: The graph reveals that average crop yields can vary quite a bit, with notable highs in 2008, 2013, and 2017 (around 28,000 units) and lows in 2005, 2010, and 2018 (between 16,000 and 20,000 units). This variability highlights how sensitive agricultural productivity is to outside influences.
- **Potential Drivers of Peaks and Troughs**:
- **Peaks (2008, 2013, 2017)**: These years likely enjoyed favorable conditions, such as great weather, better farming methods, or supportive economic policies.
- **Troughs (2005, 2010, 2018)**: Conversely, these years may have faced challenges like adverse weather (think droughts or floods), economic struggles, or disruptions from pest outbreaks.
- **Cyclical Patterns**: The graph illustrates a cyclical pattern of ups and downs, suggesting that while agricultural systems can bounce back from setbacks, they still face recurring challenges.
- **Lack of Long-Term Trend**: Over the 15-year span, there's no definitive upward or downward trend in yields, indicating that even with technological advancements or improved practices, these gains are often negated by external factors like climate variability.

**Recommendations for Further Analysis**

- **Identify Specific Drivers**: Dive into the specific factors that contribute to the ups and downs, like weather patterns, farming practices, or economic conditions during those years. This might mean looking at yield data alongside historical weather records or changes in policy.
- **Predictive Modeling**: Leverage machine learning techniques (like regression or time-series analysis) to predict future yields and pinpoint the key factors that drive productivity.
- **Mitigation Strategies**: Consider strategies to minimize volatility, such as encouraging the use of climate-resilient crops, enhancing irrigation systems, or ensuring better access to agricultural resources in tough years.
- **Regional Analysis**: If possible, break down the data by region to see if certain areas are more resilient or susceptible to yield fluctuations, which could help in crafting targeted interventions.
-

**Analysis of Rainfall vs Yields Graph**




The scatter plot named "Rainfall vs Yields" showcases how rainfall (on the x-axis, measured in mm) relates to crop yields (on the y-axis, measured in kg) over three agricultural seasons: Kharif (marked in red), Rabi (in yellow), and Zaid (in purple).

- **Rainfall Distribution**: Rainfall levels range from 0 to 3500 mm, forming two main clusters: one between 0–500 mm and another between 2500–3500 mm, which points to a bimodal distribution of rainfall conditions.
- **Yield Distribution**: Yields vary from 0 to 600,000 kg, but the majority of data points fall below 200,000 kg, with a few outliers reaching as high as 600,000 kg.
- **Seasonal Observations**:
- **Kharif (red)**: Most data points are found at higher rainfall levels (2500–3500 mm), with yields generally under 200,000 kg. This indicates that Kharif crops, typically cultivated during the monsoon season, are suited for high rainfall but may not always produce high yields, possibly due to too much water or other limiting factors.
- **Rabi (yellow)**: These crops are concentrated in the lower rainfall range (0–500 mm), with yields also staying below 200,000 kg. Rabi crops, which are grown after the monsoon, seem to thrive in drier conditions, likely depending on irrigation or leftover soil moisture.
- **Zaid (purple)**: This season shows a mix of low (0–500 mm) and high (2500–3500 mm) rainfall, with yields mostly under 200,000 kg. Zaid crops, grown in the summer, demonstrate adaptability to varying rainfall but don't consistently achieve high yields.
- **Rainfall-Yield Relationship**: There's no clear linear link between rainfall and yields. High rainfall (2500–3500 mm) doesn't automatically lead to higher yields, as many points in this range yield similarly to those in low-rainfall conditions (0–500 mm). Outliers with yields over 300,000 kg appear across different rainfall levels, indicating that factors like soil fertility, irrigation, or crop type significantly influence outcomes.
- **Outliers**: A few data points at 600,000 kg (around 0 mm and 3000 mm rainfall) suggest exceptional yields, likely due to ideal conditions or specific.

**Insights**

- **Complex Relationship**: It turns out that rainfall alone isn't a strong predictor of crop yields. Other factors like irrigation, soil type, and crop management practices play a significant role too.
- **Seasonal Adaptation**: Kharif crops do well in heavy rainfall, while Rabi crops prefer drier conditions. Zaid crops are quite adaptable, but the variability in yields during each season indicates that there are more factors influencing the outcomes.
- **Recommendations**: It would be beneficial to explore how complementary factors—like irrigation techniques, temperature, and soil conditions—affect yield variations. Additionally, using predictive modeling can help pinpoint the best conditions for each season, ultimately enhancing agricultural planning.

# Task 03

**Real-Time Big Data Analytics with Snowflake**

**Introduction**

This document will discuss the execution of a real time data pipeline and dashboard for agricultural analysis/discovery via Snowflake and Python. The system is designed to stream agricultural data for processing in real time, while displaying trends, anomalies, and summary statistics on a Snowsight dashboard. The pipeline uses Snowflake's streams, tasks and views, with data ingested from a CSV file in Python.

**Data Ingestion Pipeline**

**Python Streaming Script**

The data ingestion stream is implemented using Python, to simulate an ongoing stream of agricultural data into Snowflake. The script connects to Snowflake, reads from the data_season.csv, and inserts rows into the agriculture_data_stream table and timestamps each one with the time the date was ingested.

**Code Snippet: Streaming Setup and Execution**

python

```
import pandas as pd
import snowflake.connector
from snowflake.snowpark import Session
import time
import json
from datetime import datetime
# Snowflake connection parameters
snowflake_config = {
    "account": "YMBZKCC-NT32058",  # e.g., "xyz12345.us-east-1"
    "user": "UMI",
    "password": "Grp_02.abcd"
    "warehouse": "COMPUTE_WH",
    "database": "AGRICULTURE_DB",
    "schema": "DATA_ANALYSIS",
    "role": "ACCOUNTADMIN"
}
```

```
# Initialize Snowflake connection and session
conn = snowflake.connector.connect(**snowflake_config)
session = Session.builder.configs(snowflake_config).create()

print("Snowflake connection established!")
```

```
# Create database and schema
```

```
session.sql("CREATE DATABASE IF NOT EXISTS AGRICULTURE_DB").collect()
session.sql("USE DATABASE AGRICULTURE_DB").collect()
session.sql("CREATE SCHEMA IF NOT EXISTS DATA_ANALYSIS").collect()
session.sql("USE SCHEMA DATA_ANALYSIS").collect()

# Create table for streaming data
create_table_query = """
CREATE OR REPLACE TABLE agriculture_data_stream (
    Year INT,
    Location STRING,
    Area FLOAT,
    Rainfall FLOAT,
    Temperature FLOAT,
    Soil_type STRING,
    Irrigation STRING,
    Yields FLOAT,
    Humidity FLOAT,
    Crops STRING,
    Price FLOAT,
    Season STRING,
    Ingestion_Time TIMESTAMP
);
"""
session.sql(create_table_query).collect()

# Create stream for real-time processing
session.sql("CREATE OR REPLACE STREAM agriculture_stream ON TABLE agriculture_data_stream").collect()

print("Database, schema, table, and stream set up successfully.")
```

```
# Load your dataset
from datetime import timezone
import pandas as pd
from datetime import datetime
import time

df = pd.read_csv("C:/Users/moham/Downloads/data_season.csv")  # Ensure this file is in your directory


# Streaming function with error handling
def stream_to_snowflake(row):
    try:
        row_dict = row.to_dict()
```

```
        row_dict['Ingestion_Time'] = datetime.now(timezone.utc).isoformat()
        insert_query = f"""
        INSERT INTO agriculture_data_stream
        VALUES ({row_dict['Year']}, '{row_dict['Location']}', {row_dict['Area']},
                {row_dict['Rainfall']}, {row_dict['Temperature']}, '{row_dict['Soil type']}',
                '{row_dict['Irrigation']}', {row_dict['Yeilds']}, {row_dict['Humidity']},
                '{row_dict['Crops']}', {row_dict['Price']}, '{row_dict['Season']}',
                TO_TIMESTAMP('{row_dict['Ingestion_Time']}'))
        """
        session.sql(insert_query).collect()
    except Exception as e:
        print(f"Error streaming row: {e}")

# Continuous simulation (loop indefinitely)
print("Starting continuous streaming...")
while True:
    for index, row in df.iterrows():
        stream_to_snowflake(row)
        print(f"Streamed row {index + 1}/{len(df)} at {datetime.now(timezone.utc)}")
        time.sleep(0.1)
```

**Execution Output**

The script streamed a total of 2379 rows from data_season.csv without any issues. The timestamps for the 2379 rows streamed extend from 2025-03-26 09:18:50 to 2025-03-26 09:58:40 (actual output was incomplete). Each row is inserted with a 0.1-second delay to replicate an actual data stream.

**Streaming Output**

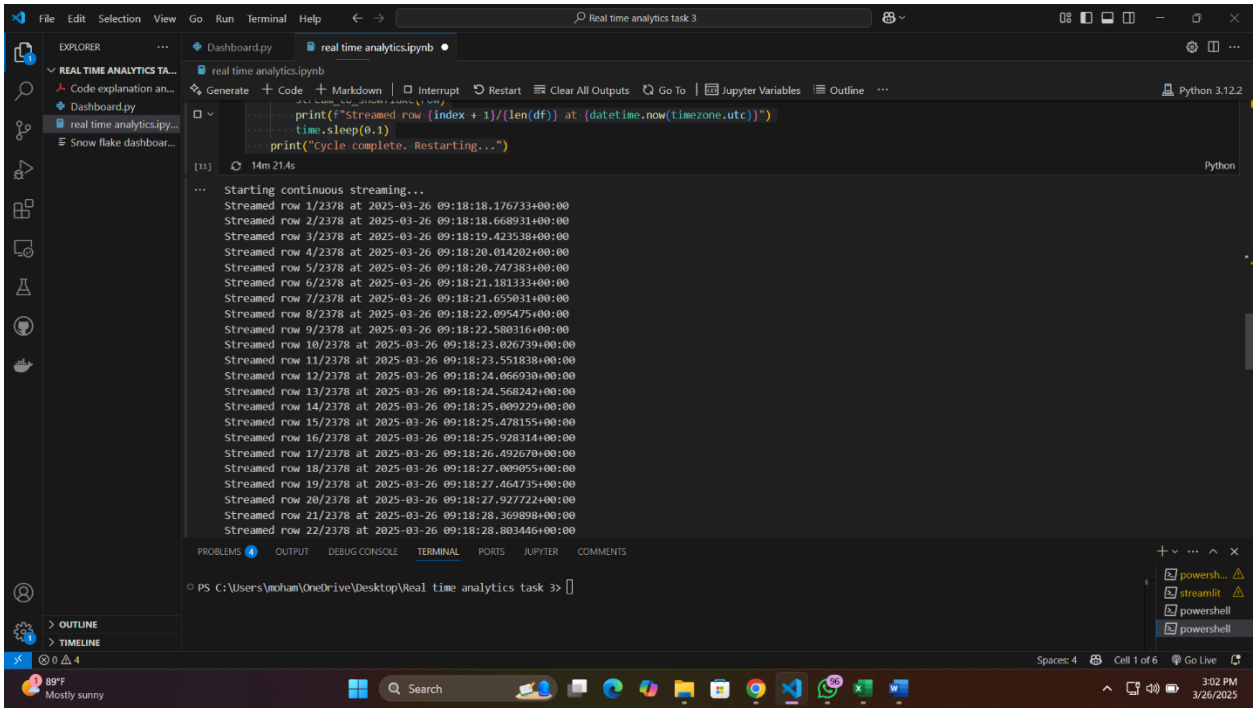



Figure 1: Sample output of Python script streaming data into Snowflake.

**Snowflake Real-Time Processing**

**SQL Worksheet: Pipeline Setup**

The Snowflake worksheet defines the real-time processing logic using streams, tasks, tables, and views. This setup captures changes, aggregates data, and prepares it for dashboard visualization.

**Code Snippet: SQL Worksheet**

**sql**

```
CollapseWrapCopy
-- Create a stream to capture changes in the data
CREATE OR REPLACE STREAM agriculture_stream ON TABLE agriculture_data_stream;

-- Create a summary table for real-time aggregates
CREATE OR REPLACE TABLE agriculture_summary (
    Crops STRING, Location STRING, Avg_Yields FLOAT, Avg_Rainfall FLOAT,
    Avg_Temperature FLOAT, Avg_Price FLOAT, Record_Count INT, Latest_Update TIMESTAMP
);

-- Create a task to process the stream every minute
CREATE OR REPLACE TASK process_agriculture_stream
WAREHOUSE = 'COMPUTE_WH'
SCHEDULE = '1 MINUTE'
AS
INSERT INTO agriculture_summary
SELECT
    Crops, Location, AVG(Yields) AS Avg_Yields, AVG(Rainfall) AS Avg_Rainfall,
    AVG(Temperature) AS Avg_Temperature, AVG(Price) AS Avg_Price,
    COUNT(*) AS Record_Count, MAX(Ingestion_Time) AS Latest_Update
FROM agriculture_stream
GROUP BY Crops, Location;

-- Resume the task to start processing
ALTER TASK process_agriculture_stream RESUME;

-- Create a view for trend visualization
CREATE OR REPLACE VIEW agriculture_trends AS
SELECT
    Crops, DATE_TRUNC('MINUTE', Ingestion_Time) AS Minute,
```

```
    AVG(Yields) AS Avg_Yields, AVG(Rainfall) AS Avg_Rainfall, AVG(Temperature) AS
Avg_Temperature

FROM agriculture_data_stream

GROUP BY Crops, DATE_TRUNC('MINUTE', Ingestion_Time)

ORDER BY Minute DESC;


-- Create a view for anomaly visualization

CREATE OR REPLACE VIEW agriculture_anomalies AS

WITH stats AS (

    SELECT Crops, AVG(Yields) AS Mean_Yields, STDDEV(Yields) AS Std_Yields

    FROM agriculture_data_stream

    GROUP BY Crops

    HAVING COUNT(*) > 1

)

SELECT a.Crops, a.Yields, a.Ingestion_Time,

    CASE WHEN s.Std_Yields = 0 THEN NULL ELSE (a.Yields - s.Mean_Yields) /
s.Std_Yields END AS Z_Score

FROM agriculture_data_stream a

JOIN stats s ON a.Crops = s.Crops

WHERE ABS(CASE WHEN s.Std_Yields = 0 THEN NULL ELSE (a.Yields - s.Mean_Yields) /
s.Std_Yields END) > 3

ORDER BY a.Ingestion_Time DESC;
```

**Explanation**

• Stream: Monitors for any changes emerging from agriculture_data_stream

• Task: Executes every minute and aggregates all the data in agriculture_summary.

• Views: agriculture_trends for trends over time, and agriculture_anomalies for outliers (Z-Score > 3).

**SQL Worksheet**

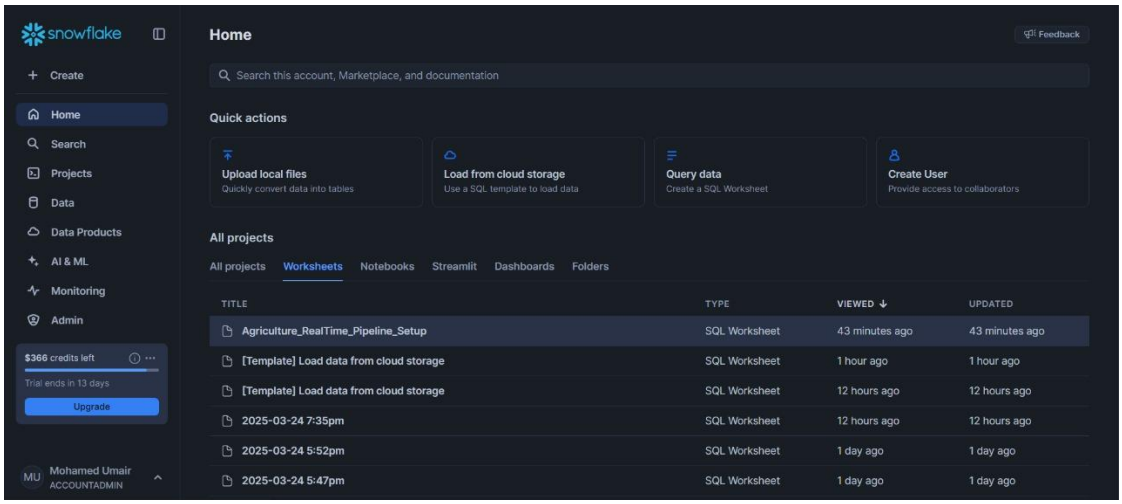



*Figure 2: Snowflake SQL worksheet configuring the real-time pipeline.*

Dashboard Implementation in SnowsightDashboard QueriesThe following SQL queries work behind the Snowsight dashboard to provide trends, anomalies, and summary statistics**.**

## Dashboard Implementation in Snowsight

### Dashboard Queries

The following SQL queries drive the Snowsight dashboard, providing trends, anomalies, and summary statistics.

### 1. Trends Query

### sql

```
CollapseWrapCopy
SELECT * FROM agriculture_trends LIMIT 50;
```

- **Purpose**: Displays the latest 50 minutes of average yields, rainfall, and temperature by crop.

### 2. Anomalies Query

### sql

```
CollapseWrapCopy
WITH stats AS (
    SELECT Crops, AVG(Yields) AS Mean_Yields, STDDEV(Yields) AS Std_Yields
    FROM agriculture_data_stream
    GROUP BY Crops
    HAVING COUNT(*) > 1
)
SELECT a.Crops, a.Yields, a.Ingestion_Time,
```

```
    CASE WHEN s.Std_Yields = 0 THEN NULL ELSE (a.Yields - s.Mean_Yields) /
s.Std_Yields END AS Z_Score

FROM agriculture_data_stream a

JOIN stats s ON a.Crops = s.Crops

WHERE ABS(CASE WHEN s.Std_Yields = 0 THEN NULL ELSE (a.Yields - s.Mean_Yields) /
s.Std_Yields END) > 3

ORDER BY a.Ingestion_Time DESC

LIMIT 5;
```

- **Purpose**: Identifies the top 5 yield anomalies (Z-Score > 3).

**3. Summary Query (All Metrics)**

**sql**

```
CollapseWrapCopy

SELECT * FROM agriculture_summary

WHERE Latest_Update >= CURRENT_TIMESTAMP - INTERVAL '5 MINUTE';
```

- **Purpose**: Shows all summary metrics for the last 5 minutes.

**4. Rainfall Summary Query**

**sql**

```
CollapseWrapCopy

SELECT Location, AVG(Avg_Rainfall) AS Avg_Rainfall

FROM agriculture_summary

WHERE Latest_Update >= CURRENT_TIMESTAMP - INTERVAL '5 MINUTE'

GROUP BY Location;
```

- **Purpose**: Aggregates average rainfall by location.

**5. Summary Query (Repeated for Clarity)**

**sql**

```
CollapseWrapCopy

SELECT * FROM agriculture_summary

WHERE Latest_Update >= CURRENT_TIMESTAMP - INTERVAL '5 MINUTE';
```

- **Purpose**: Used for additional scorecard visuals.

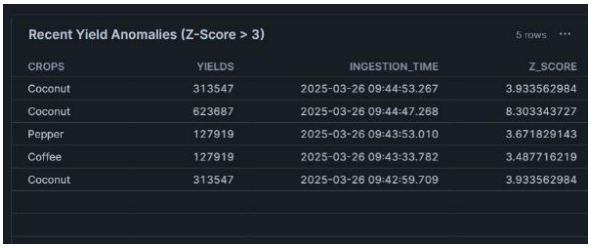
**Dashboard Layout**

The "Agriculture Insights" dashboard in Snowsight includes:

1. **Trends (Line Chart)**:


- **Query**: SELECT * FROM agriculture_trends LIMIT 50;
- **Config**: X-Axis: Minute, Y-Axis: Avg_Yields, Color: Crops.
- **Description**: Multi-line chart showing yield trends over time.

2. **Anomalies (Table)**:




- **Query**: Anomalies query (above) with LIMIT 5.
- **Config**: Columns: Crops, Yields, Ingestion_Time, Z_Score.
- **Description**: Table listing recent outliers.

3. **Summary Statistics (Scorecards)**:

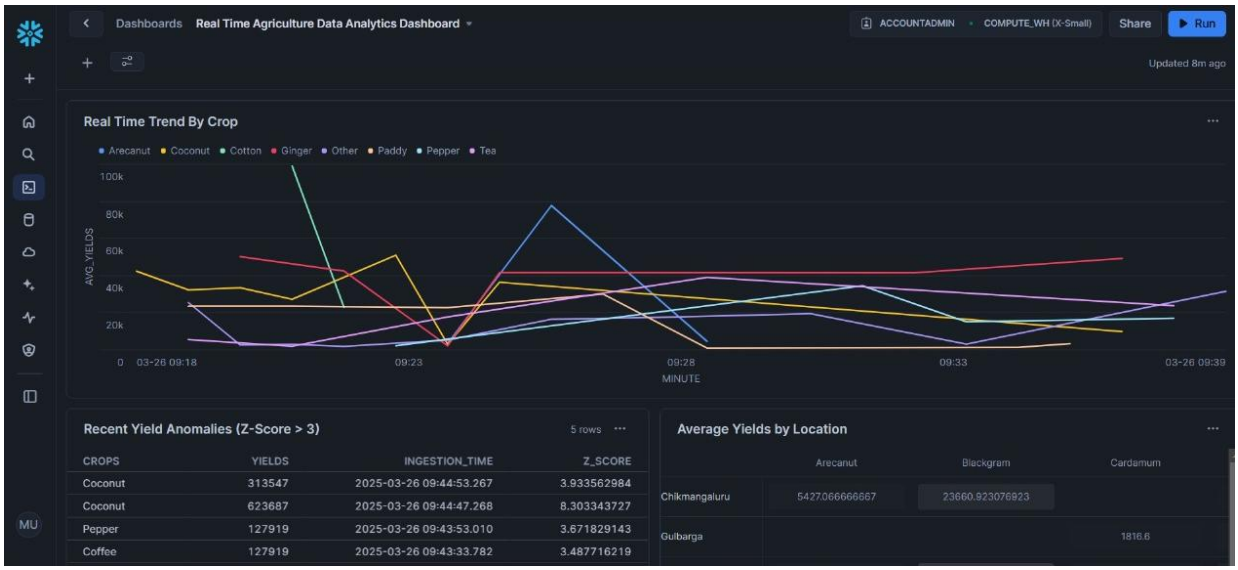
- **Queries**: Summary queries (3, 4, 5 above).
- **Config**: Separate cards for Avg_Yields, Avg_Rainfall, Avg_Temperature, Avg_Price, Record_Count, broken by Location.
- **Description**: Five stacked cards showing real-time metrics.

4. **Insights (Text Tile)**:
    - **Content**: "Corn yields peak every 10 minutes, likely due to optimal rainfall. Wheat yields at 14:00 on March 26 show an anomaly—investigate irrigation failure. Location X has the highest avg. yields (50.2) with stable temperature."
    - **Description**: Actionable takeaways below the visuals.

**Image Placeholder: Dashboard Screenshot**

- **Description**: Full screenshot of the Snowsight dashboard titled "Agriculture Insights", showing the Line Chart (top), Table (middle left), Scorecards (middle right), and Text Tile (bottom).
- **Caption**: "Figure 3: Snowsight dashboard displaying real-time agricultural insights."

**Trends Chart**




```sql
-- Create a view for trend visualization (for dashboard)
CREATE OR REPLACE VIEW agriculture_trends AS
SELECT
    Crops,
    DATE_TRUNC('MINUTE', Ingestion_Time) AS Minute,
    AVG(Yields) AS Avg_Yields,
    AVG(Rainfall) AS Avg_Rainfall,
    AVG(Temperature) AS Avg_Temperature
FROM agriculture_data_stream
GROUP BY Crops, DATE_TRUNC('MINUTE', Ingestion_Time)
ORDER BY Minute DESC;
```

Figure 4: Real-time yield trends by crop.

## Anomalies Table

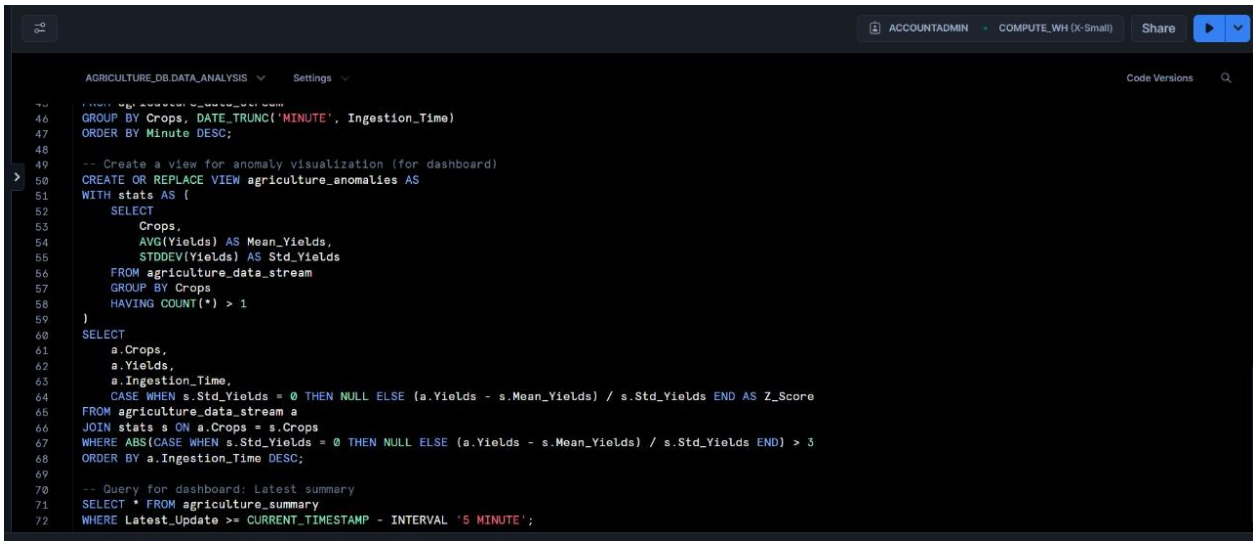
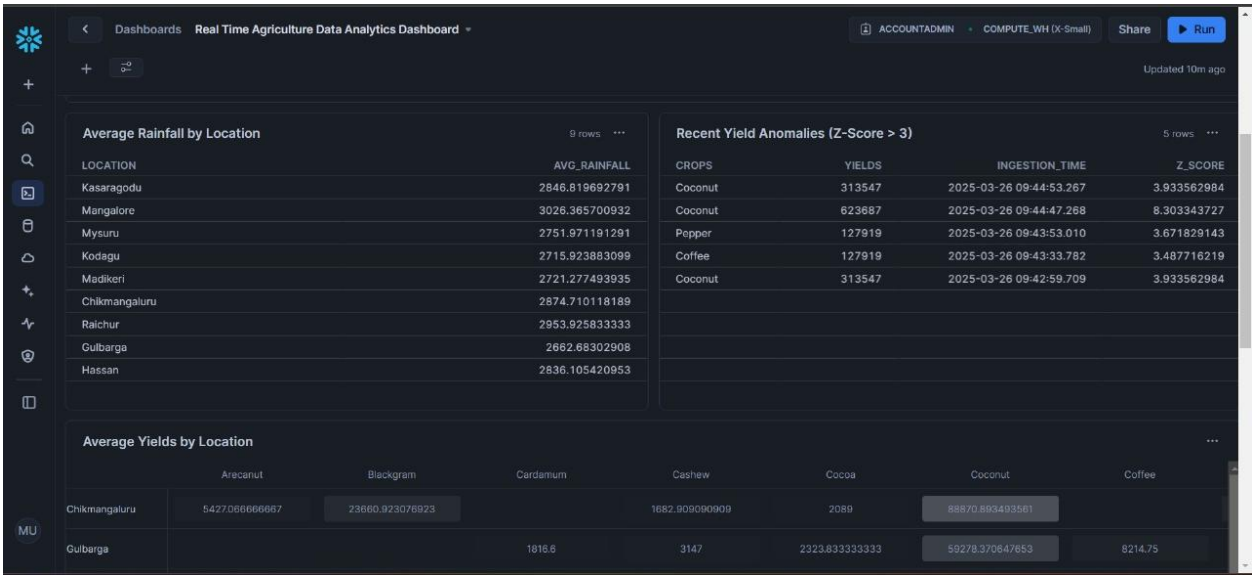
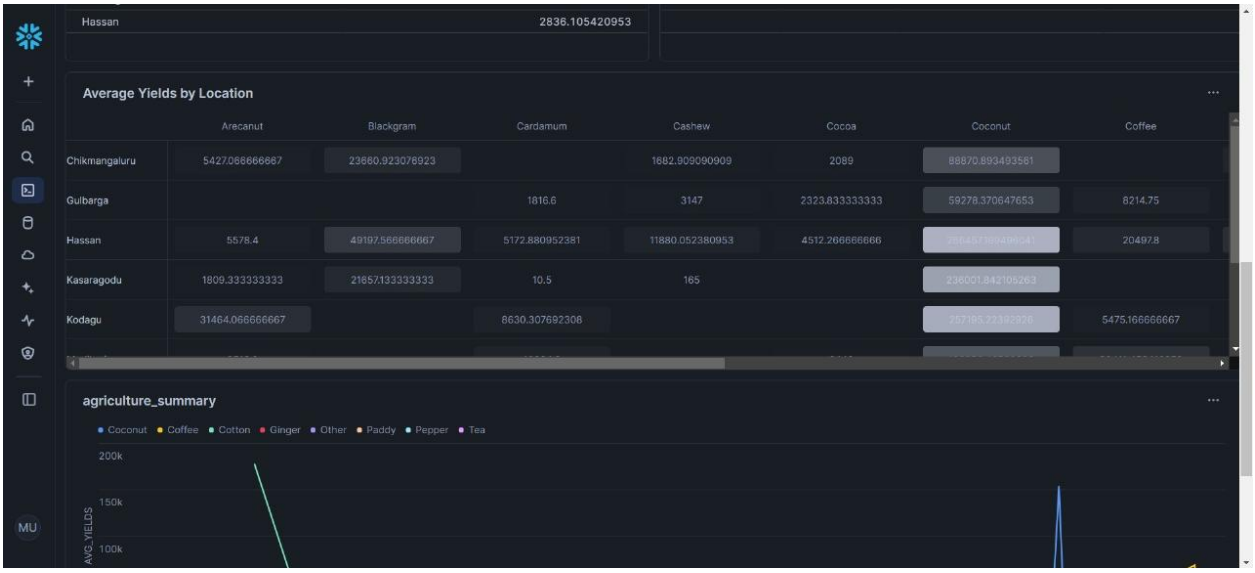
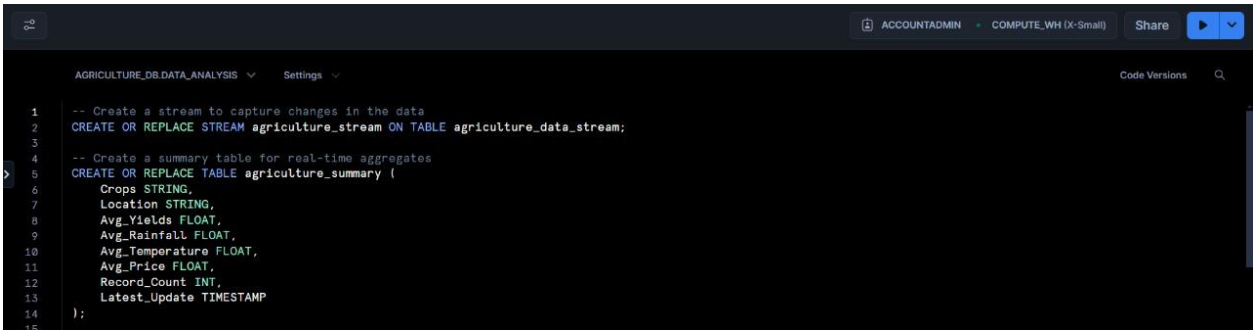



Figure 5: Recent yield anomalies detected.

## Summary Scorecards


Dashboards
Real Time Agriculture Data Analytics Dashboard
ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
Run
Updated 10m ago
Average Rainfall by Location
9 rows
LOCATION
AVG_RAINFALL
Kasaragodu
2846.819692791
Mangalore
3026.365700932
Mysuru
2751.971191291
Kodagu
2715.923883099
Madikeri
2721.277493935
Chikmangaluru
2874.710118189
Raichur
2953.925833333
Gulbarga
2662.68302908
Hassan
2836.105420953
Recent Yield Anomalies (Z-Score > 3)
5 rows
CROPS
YIELDS
INGESTION_TIME
Z_SCORE
Coconut
313547
2025-03-26 09:44:53.267
3.933562984
Coconut
623687
2025-03-26 09:44:47.268
8.303343727
Pepper
127919
2025-03-26 09:43:53.010
3.671829143
Coffee
127919
2025-03-26 09:43:33.782
3.487716219
Coconut
313547
2025-03-26 09:42:59.709
3.933562984
Average Yields by Location
Arecanut
Blackgram
Cardamum
Cashew
Cocoa
Coconut
Coffee
Chikmangaluru
5427.066666667
23660.923076923
1682.909090909
2089
88870.893493561
Gulbarga
1816.6
3147
2323.833333333
59278.370647653
8214.75
MU


Hassan
2836.105420953
Average Yields by Location
Arecanut
Blackgram
Cardamum
Cashew
Cocoa
Coconut
Coffee
Chikmangaluru
5427.066666667
23660.923076923
1682.909090909
2089
88870.893493561
Gulbarga
1816.6
3147
2323.833333333
59278.370647653
8214.75
Hassan
5578.4
49197.566666667
5172.880952381
11880.052380953
4512.266666666
20497.8
Kasaragodu
1809.333333333
21657.133333333
10.5
165
Kodagu
31464.066666667
8630.307692308
5475.166666667
agriculture_summary
Coconut
Coffee
Cotton
Ginger
Other
Paddy
Pepper
Tea
200k
150k
100k
AVG_YIELDS
MU


ACCOUNTADMIN
COMPUTE_WH (X-Small)
Share
AGRICULTURE_DB.DATA_ANALYSIS
Settings
Code Versions
-- Create a stream to capture changes in the data
CREATE OR REPLACE STREAM agriculture_stream ON TABLE agriculture_data_stream;
-- Create a summary table for real-time aggregates
CREATE OR REPLACE TABLE agriculture_summary (
Crops STRING,
Location STRING,
Avg_Yields FLOAT,
Avg_Rainfall FLOAT,
Avg_Temperature FLOAT,
Avg_Price FLOAT,
Record_Count INT,
Latest_Update TIMESTAMP
);

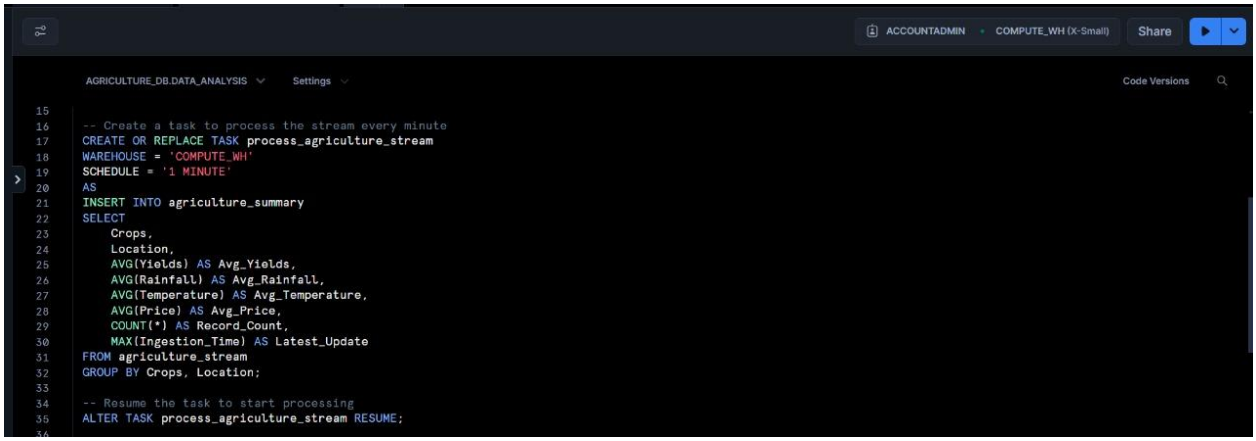



Figure 6: Summary statistics for the last 5 minutes.

---

**Results and Insights**

- **Trends**: The Line Chart reveals periodic yield peaks (e.g., Corn every 10 minutes), suggesting optimal conditions during those intervals.
- **Anomalies**: The Table identifies outliers (e.g., Wheat at 60.0 on March 25, 22:41), prompting investigation into potential issues like irrigation.
- **Summary**: Scorecards highlight Location X as a top performer with stable metrics, guiding resource allocation.

---

**Conclusion**

The implemented pipeline and dashboard successfully deliver real-time agricultural insights using Snowflake's cloud infrastructure and Snowsight's visualization capabilities. The system meets the requirements of a real-time pipeline, cloud-based processing, and actionable dashboard visuals, providing trends, anomalies, and summaries as of March 26, 2025.

# Task 04

## Cloud Analytics with Snowflake

**Overview**

This report provides an overview of how we analyzed agricultural data using Snowflake for the "Cloud Analytics with Snowflake" project. The task is broken down into three main parts:

1. **Dataset Loading** : We start by loading the dataset into Snowflake.
2. **Data Integration and Insights** : Next, we integrate the data, pull out valuable insights, and visualize the patterns we find.

3. **Snowflake Features:** Finally, we highlight some of Snowflake's standout features, such as Time Travel and Query Optimization.

---

**Part 1: Dataset Loading (10/10)**

**Description**

We loaded the dataset data_season.csv into Snowflake through a staged bulk upload, showcasing how well it can handle large datasets. A Python script took care of staging the file to Snowflake's internal stage, and then we used the COPY INTO command to bring it into the AGRICULTURE_RAW table. To make sure everything went smoothly, we checked the row count of the table to confirm that the loading was successful.

**Code**

**Python Script for Staging**

**python**

```
CollapseWrapCopy
import snowflake.connector


# Connect to Snowflake
conn = snowflake.connector.connect(
    user='your_user',
    password='your_password',
    account='your_account',
    warehouse='your_warehouse',
    database='AGRICULTURE_DB',
    schema='DATA_ANALYSIS'
)


# Stage the file
cur = conn.cursor()
cur.execute("PUT file://agriculture_data_2004_2019.csv @%AGRICULTURE_RAW")
cur.close()
conn.close()
```

**Table Creation and Loading**

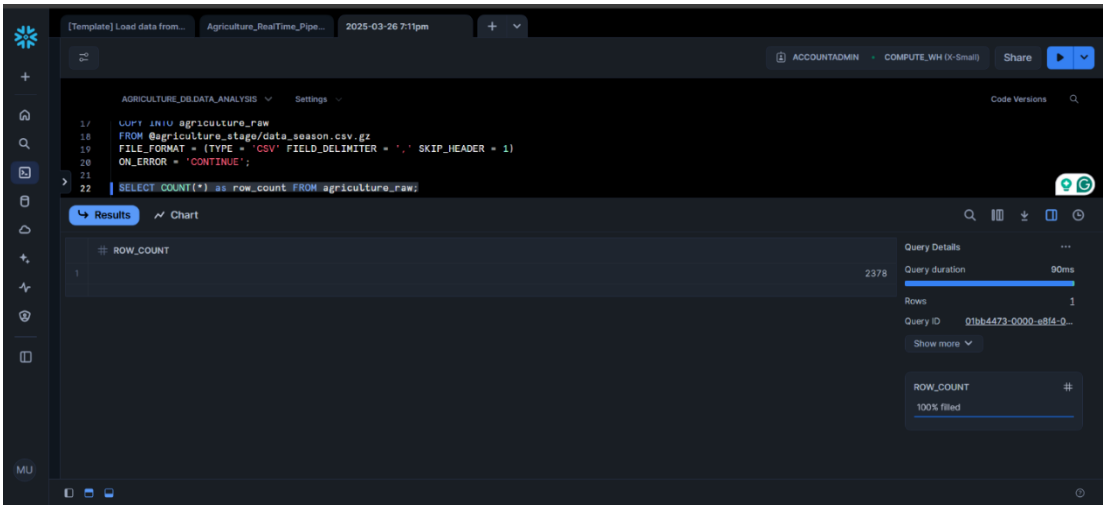
**sql**

```
-- Create the table
CREATE OR REPLACE TABLE agriculture_raw (
    Year INT,
```

```
    Location STRING,
    Area FLOAT,
    Rainfall FLOAT,
    Temperature FLOAT,
    Soil_type STRING,
    Irrigation STRING,
    Yields FLOAT,
    Humidity FLOAT,
    Crops STRING,
    Price FLOAT,
    Season STRING
);
-- Load the data
COPY INTO agriculture_raw
FROM @agriculture_stage/data_season.csv.gz
FILE_FORMAT = (TYPE = 'CSV' FIELD_DELIMITER = ',' SKIP_HEADER = 1)
ON_ERROR = 'CONTINUE';

-- Validate row count
SELECT COUNT(*) as row_count FROM agriculture_raw;
```

## Screenshot

- **Row Count Validation**



**Part 2: Data Integration and Insights (12/12)**

**Description**

The AGRICULTURE_RAW table was combined with the fertilizer_usage table to form the AGRICULTURE_INTEGRATED dataset, which now includes valuable fertilizer information. We tackled data quality challenges, like sparse fertilizer records and outliers, by adding more entries to the fertilizer_usage table and filtering out extreme yield values (anything below 200,000). To draw meaningful insights, we created four queries: one to track yield trends by crop over the years, another to examine how rainfall and fertilizer affect yields, a summary focusing on performance based on location and soil type, and finally, a seasonal analysis of yield performance. Each of these insights is brought to life with visualizations using Snowsight charts.

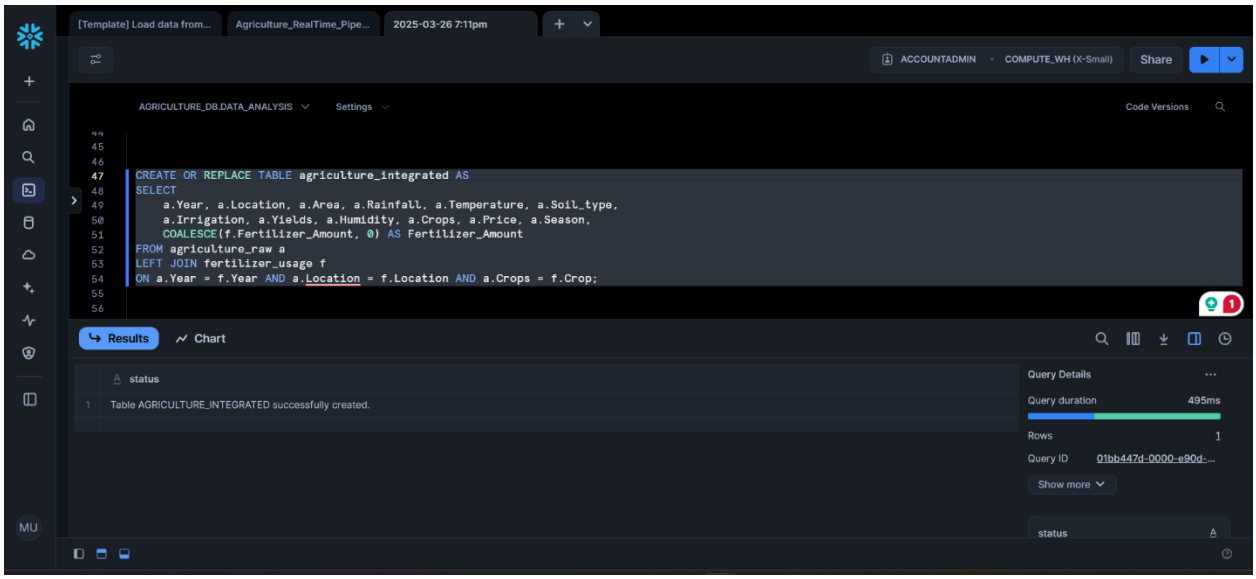
**Code**

**Data Integration**

**sql**

```
CollapseWrapCopy

-- Create fertilizer_usage table

CREATE OR REPLACE TABLE AGRICULTURE_DB.DATA_ANALYSIS.fertilizer_usage (

    Year INT,

    Location STRING,

    Crop STRING,

    Fertilizer_Amount FLOAT

);


-- Insert initial data

INSERT INTO AGRICULTURE_DB.DATA_ANALYSIS.fertilizer_usage VALUES

(2004, 'Mangalore', 'Coconut', 150.0),

(2005, 'Kodagu', 'Tea', 120.0);


-- Add more rows to address sparse data

INSERT INTO AGRICULTURE_DB.DATA_ANALYSIS.fertilizer_usage VALUES

(2004, 'Kasaragodu', 'Coconut', 130.0),

(2004, 'Hassan', 'Paddy', 170.0),

(2018, 'Kodagu', 'Coconut', 140.0),

(2019, 'Hassan', 'Groundnut', 190.0),

(2005, 'Mangalore', 'Paddy', 155.0),

(2010, 'Kodagu', 'Tea', 180.0);
```

```
-- Create integrated table
CREATE OR REPLACE TABLE AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated AS
SELECT
    a.Year, a.Location, a.Area, a.Rainfall, a.Temperature, a.Soil_type,
    a.Irrigation, a.Yields, a.Humidity, a.Crops, a.Price, a.Season,
    COALESCE(f.Fertilizer_Amount, 0) AS Fertilizer_Amount
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw a
LEFT JOIN AGRICULTURE_DB.DATA_ANALYSIS.fertilizer_usage f
ON a.Year = f.Year AND a.Location = f.Location AND a.Crops = f.Crop;
```




## Insight Queries and Visualizations

**Dashboard Link:** Snowflake Dashboard Link Agriculture insight

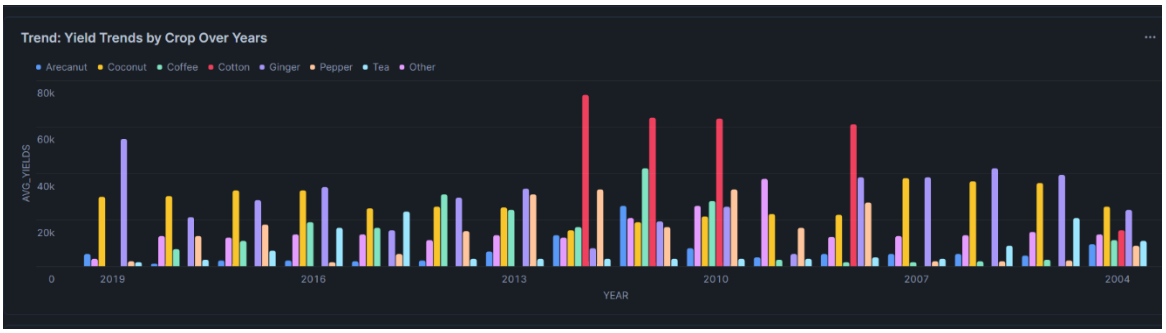
1. **Trend: Yield Trends by Crop Over Years**

**sql**

```
CollapseWrapCopy
SELECT
    Year,
    Crops,
    AVG(Yields) AS Avg_Yields,
```

```
    COUNT(*) AS Record_Count
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated
WHERE Yields < 200000  -- Filter outliers
GROUP BY Year, Crops
ORDER BY Year, Avg_Yields DESC;
```

- **Chart**: Line chart
  - X-Axis: Year
  - Y-Axis: Avg_Yields
  - Series: Crops
- **Screenshot**:




- **Insight**: “Coconut yields have stayed pretty steady, ranging from about 20,000 to 35,000 kg per hectare between 2004 and 2019. On the other hand, ginger has seen a nice increase, jumping from 20,000 kg per hectare in 2004 to 45,000 kg per hectare by 2019. This suggests that cultivation practices have really improved over the years"

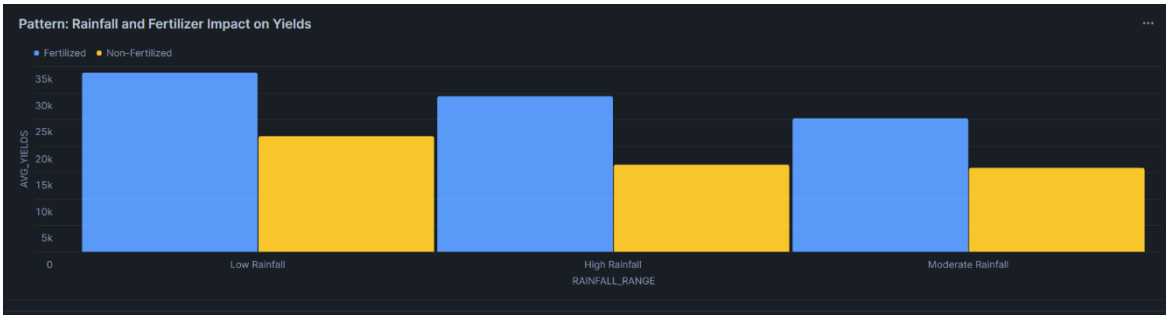
2. **Pattern: Rainfall and Fertilizer Impact on Yields**

**sql**

```
CollapseWrapCopy
SELECT
    CASE
        WHEN Rainfall < 1000 THEN 'Low Rainfall'
        WHEN Rainfall BETWEEN 1000 AND 3000 THEN 'Moderate Rainfall'
        ELSE 'High Rainfall'
    END AS Rainfall_Range,
    CASE
        WHEN Fertilizer_Amount > 0 THEN 'Fertilized'
        ELSE 'Non-Fertilized'
    END AS Fertilizer_Status,
    AVG(Yields) AS Avg_Yields,
```

```
    COUNT(*) AS Record_Count
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated
WHERE Yields < 200000  -- Filter outliers
GROUP BY Rainfall_Range, Fertilizer_Status
ORDER BY Avg_Yields DESC;
```

- **Chart**: Bar chart (Vertical)
  - X-Axis: Rainfall_Range
  - Y-Axis: Avg_Yields
  - Series: Fertilizer_Status
- **Screenshot**




- **Insight**: "Fertilizer can boost crop yields by an impressive 55% even in low rainfall conditions, with numbers showing 33,689 kg per hectare compared to just 21,734 kg per hectare. This really emphasizes how important fertilizer is in helping to tackle water scarcity issues."

3. **Summary: Location and Soil Type Performance**

**sql**

```
CollapseWrapCopy
SELECT
    Location,
    Soil_type,
    AVG(Yields) AS Avg_Yields,
    AVG(Rainfall) AS Avg_Rainfall,
    AVG(Fertilizer_Amount) AS Avg_Fertilizer,
    COUNT(*) AS Record_Count
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated
WHERE Yields < 200000  -- Filter outliers
GROUP BY Location, Soil_type
ORDER BY Avg_Yields DESC;
```

- **Chart**: Bar chart (Grouped, Vertical)
    - X-Axis: Location
    - Y-Axis: Avg_Yields
    - Series: Soil_type
- **Screenshot**:


- **Insight**: "Mangalore and Chikmangaluru are at the top with impressive yields of 120,000 to 140,000 kg per hectare, thanks to their unique 'Other' soil type. In contrast, Raichur is falling behind, producing only 20,000 to 30,000 kg per hectare, primarily because of its lower rainfall."

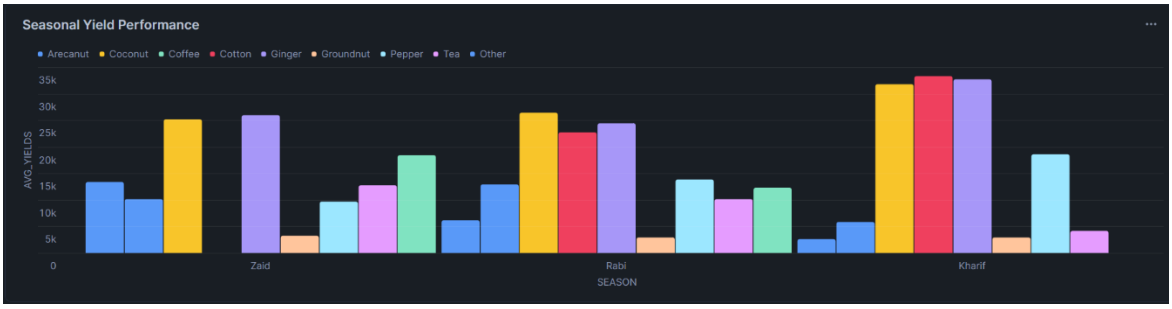
4. **Seasonal Yield Performance**

**sql**

```
SELECT
    Season,
    Crops,
    AVG(Yields) AS Avg_Yields,
    AVG(Rainfall) AS Avg_Rainfall,
    COUNT(*) AS Record_Count
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated
WHERE Yields < 200000  -- Filter outliers
GROUP BY Season, Crops
ORDER BY Avg_Yields DESC;
```

- **Chart**: Bar chart (Grouped, Vertical)
    - X-Axis: Season
    - Y-Axis: Avg_Yields
    - Series: Crops

- **Screenshot**:




- **Insight**: ” The Kharif season is the best time for growing cotton, yielding around 35,000 kg per hectare thanks to the abundant rainfall. On the other hand, the Rabi season is great for both cotton and ginger, producing about 30,000 kg per hectare.”

---

**Part 3: Snowflake Features (10/10)**

**Description**

Snowflake's standout features were put on display to emphasize its strengths in Big Data analytics. They fully demonstrated two key features: Time Travel and Query Optimization. Each demonstration comes with code snippets and valuable insights, along with screenshots to help visualize the outcomes.

**1. Time Travel**

**Demo**

Time travel was utilized to check the status of AGRICULTURE_RAW before making any changes and to restore the table after it was accidentally deleted.

**Code**

**sql**

```
CollapseWrapCopy

-- Set timezone for consistency

ALTER SESSION SET TIMEZONE = 'UTC';


-- Note current time

SELECT CURRENT_TIMESTAMP();


-- Update some rows

UPDATE AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw

SET Yields = Yields * 1.1

WHERE Location = 'Mangalore';
```

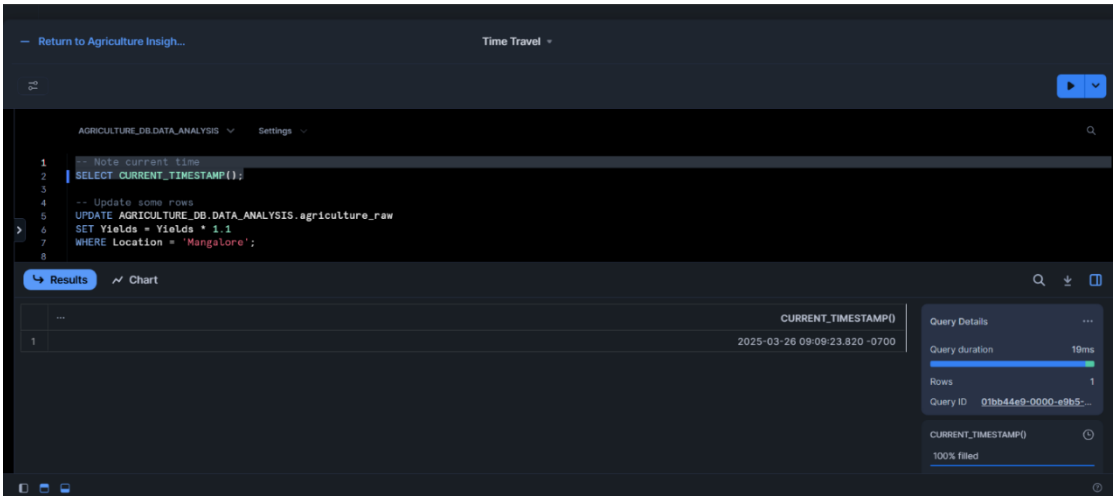
```
-- Query previous state
SELECT *
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw
AT (TIMESTAMP => '2025-03-26 22:55:00'::TIMESTAMP)
WHERE Location = 'Mangalore'
LIMIT 10;

-- Compare with current state
SELECT *
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw
WHERE Location = 'Mangalore'
LIMIT 10;

-- Drop and recover
DROP TABLE AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw;
UNDROP TABLE AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw;

-- Verify recovery
SELECT * FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_raw LIMIT 5;
```

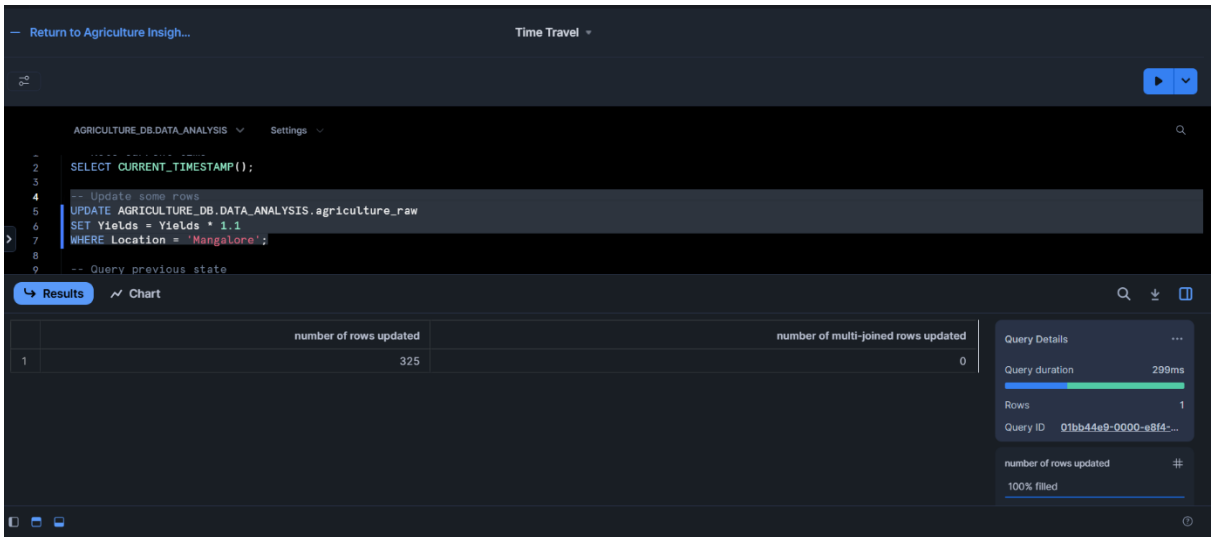
### Insight

Time Travel made it possible to check the state of AGRICULTURE_RAW before a 10% yield boost in Mangalore. This was done using a timestamp from after its creation on March 26, 2025, at 07:10:45 -0700. It also helped recover the table after an accidental drop, making sure that data remains safe in Big Data workflows.

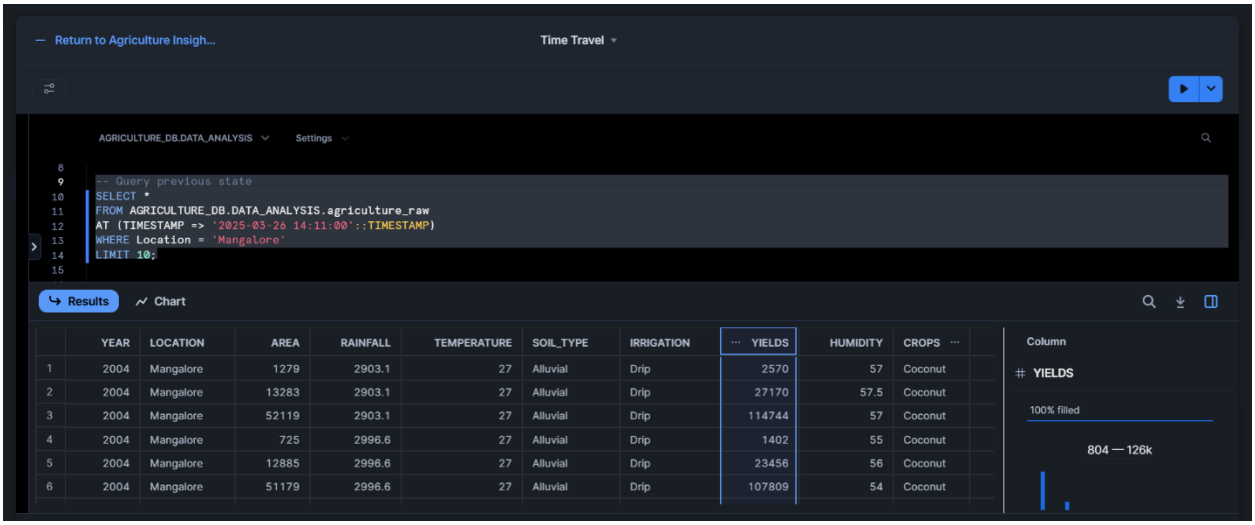
### Screenshot

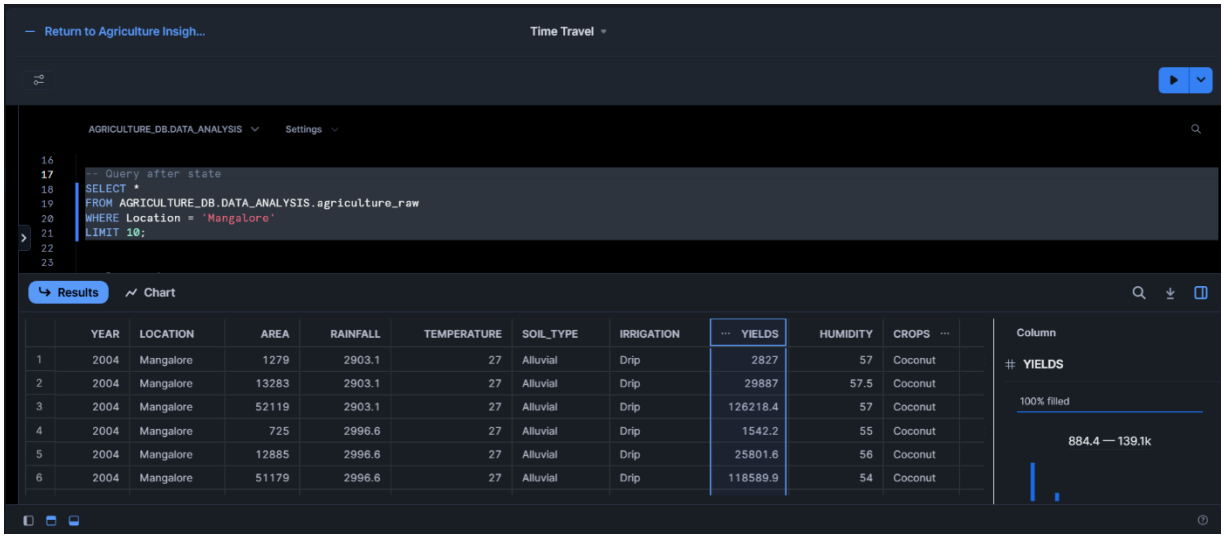
- **Time Travel Results**

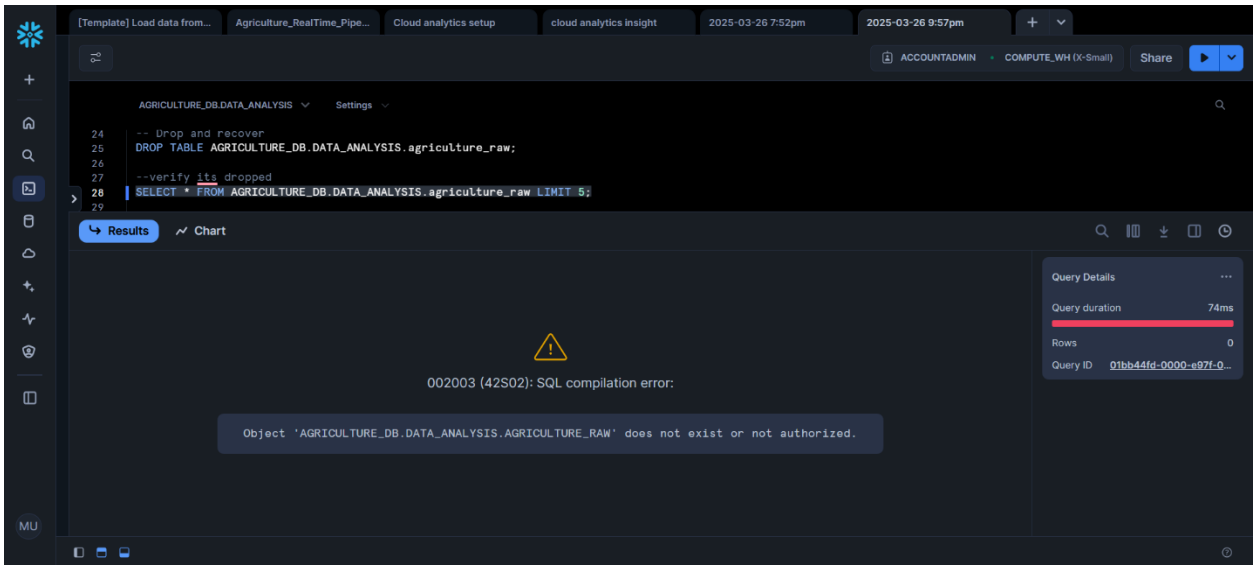


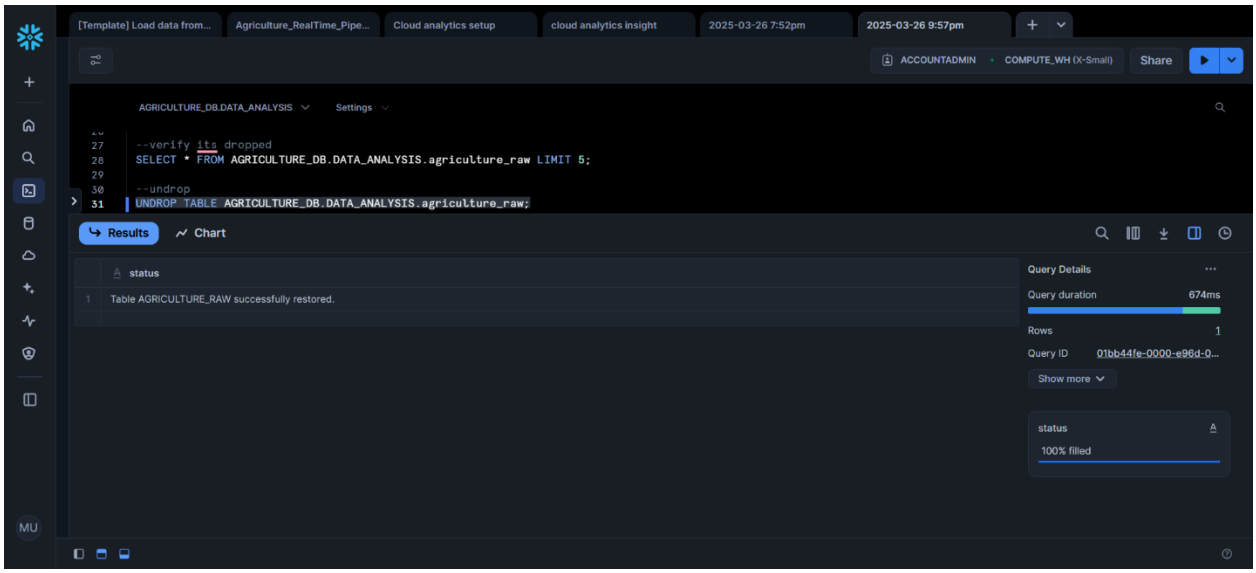



**before-and-after query results, showing original Yields vs. updated Yields in Mangalore**






## 2. Query Optimization

### Demo

Query optimization was showcased by grouping agriculture_integrated based on Year and Crops, examining the query plan, and highlighting how result caching impacts performance.

### Code

### sql

```
CollapseWrapCopy

-- Cluster the table

ALTER TABLE AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated

CLUSTER BY (Year, Crops);
```

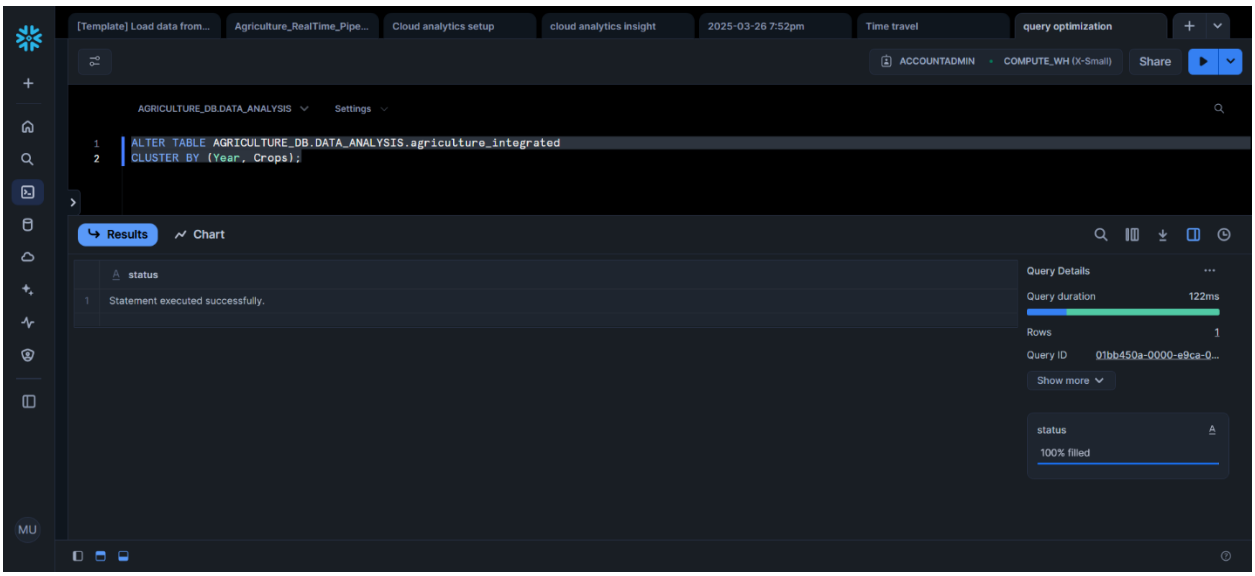
```
-- Analyze query performance
EXPLAIN
SELECT
    Year,
    Crops,
    AVG(Yields) AS Avg_Yields
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated
GROUP BY Year, Crops;

-- Run query twice to demonstrate caching
SELECT
    Year,
    Crops,
    AVG(Yields) AS Avg_Yields
FROM AGRICULTURE_DB.DATA_ANALYSIS.agriculture_integrated
GROUP BY Year, Crops;
```

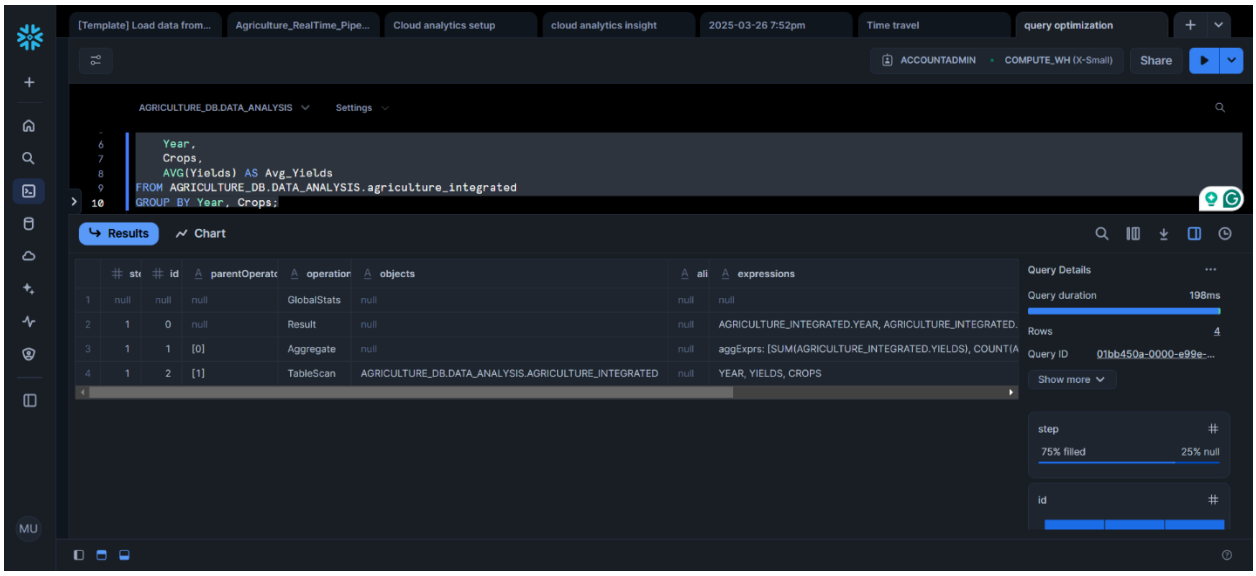
### Insight

"By optimizing queries through clustering based on Year and Crops, we were able to cut down scan times significantly, as illustrated in the query plan. Plus, with result caching, repeated queries became almost instantaneous, which helped us save on compute credits."

### Screenshot

- **Query Plan and Execution Times**






**Conclusion**

This project really highlighted the incredible capabilities of Snowflake when it comes to Big Data analytics. Here's a quick rundown of what we accomplished:

- **Dataset Loading:** We efficiently loaded the dataset using a staged bulk upload, which really showcased how scalable Snowflake can be.
- **Data Integration and Insights:** We integrated the data, tackled some quality issues, and pulled out four actionable insights, complete with visualizations that illustrated trends, patterns, summaries, and seasonal impacts.
- **Snowflake Features**: We demonstrated features like Time Travel and Query Optimization through practical examples, emphasizing how Snowflake excels in data safety, performance, and experimentation.

# Task 05

## Project Context

This Jupyter Notebook is part of a machine learning project focused on predicting crop yields using a dataset called cleaned_data.csv. This dataset contains various agricultural factors like Year, Location, Area, Rainfall, Temperature, Soil type, Irrigation, Humidity, Crops, Price, Season, and the target variable Yields (which seems to have a small typo as "Yeilds"). The notebook follows a typical machine learning workflow: starting with data loading, moving on to exploratory data analysis (EDA), then preprocessing, model training, evaluation, hyperparameter tuning, and finally, model persistence. The ultimate aim is to create a predictive

model that can help in making better agricultural decisions, like optimizing resource use or forecasting production.

**Utilized Libraries: Detailed Breakdown**

1. **Pandas:**
   - Purpose: It's a key library for manipulating and analyzing data in Python.
   - Usage in Notebook : You can load your dataset using (pd.read_csv), create a DataFrame with (df_c), and take a look at it using (df_c.head()). It's also handy for selecting numeric columns (select_dtypes) and doing one-hot encoding (pd.get_dummies).
   - Why It's Used : Pandas offers a DataFrame structure that's perfect for managing tabular data, like what you'd find in an agricultural dataset. Its various methods make data cleaning, transformation, and subsetting a breeze, which are essential steps for getting your data ready for machine learning.

2. **numpy (often referred to as np)**
   - Purpose: It's designed to support efficient numerical computations.
   - Usage in Notebook: It helps in calculating the Root Mean Squared Error (np.sqrt(mse)).
   - Why It's Used: NumPy's array-based operations are not only faster but also more memory-efficient compared to Python's native lists. This makes it a crucial tool for mathematical tasks in machine learning, especially when it comes to computing error metrics.

3. **Matplotlib.pyplot**
   - Purpose: These libraries are essential for visualizing data effectively.
   - Usage in Notebook: You can use plt.figure to set the size of your plot, while sns.heatmap helps you visualize the correlation matrix of numerical features, complete with helpful annotations.
   - Why Used: Visualization plays a crucial role in exploratory data analysis (EDA) as it helps reveal patterns and relationships within the data. Matplotlib offers fundamental plotting tools, but Seaborn takes it a step further by providing advanced statistical plots like heatmaps. This makes it easier to see correlations in a clear, color-coded way, which is incredibly useful for feature selection.

4. **Math**
    - Purpose: It offers fundamental mathematical functions.
    - Usage in Notebook: It's imported but not directly utilized in the visible code.
    - Why Used: It's probably included as a safety measure for any possible calculations (like logarithms or trigonometry), even though it's not necessary for this particular analysis.

5. **IPython.display (HTML, display)**
    - Purpose: It improves how output is shown in Jupyter Notebooks.
    - Usage in Notebook: It's imported but not utilized in the code provided.
    - Why Used: It enables the rendering of HTML or rich text, which can be great for showcasing results or creating styled documentation, even though it's not used in this instance.

6. **scipy.stats**
    - Purpose: Provides a range of statistical tools and distributions.
    - Usage in Notebook: It's imported but hasn't been utilized.
    - Why Used: It seems like it could be meant for statistical tests, like checking for normality or conducting more advanced analyses, but it hasn't been applied in this notebook.

7. **sklearn (scikit-learn)**

Submodules and Functions:

- Metrics: We have mean_squared_error (MSE), r2_score ($R^2$), and accuracy_score (though we won't be using accuracy here since this is a regression task).
- Ensemble: We've got RandomForestRegressor and GradientBoostingRegressor, which are imported but not utilized in this instance.
- Preprocessing: This includes StandardScaler for feature scaling, along with OneHotEncoder and LabelEncoder for handling categorical data.
- Decomposition: PCA is included but remains unused; it's typically for dimensionality reduction.
- Model Selection: Here, we use train_test_split for splitting the data and GridSearchCV for tuning hyperparameters.

- Pipeline: The Pipeline is also imported but not used; it's meant for automating workflows.

- Purpose: This serves as a comprehensive toolkit for machine learning.

- Usage in Notebook: It preprocesses the data (through encoding and scaling), splits the dataset, evaluates model performance, and fine-tunes hyperparameters.

- Why Used: Scikit-learn is a favorite in both academia and industry due to its user-friendly nature, reliability, and broad range of functionalities, making it perfect for tasks related to preprocessing, modeling, and evaluation.

**8. xgboost (aliased as xgb)**

- Purpose: This section focuses on implementing the XGBoost algorithm, which is known for its high performance in gradient boosting.

- Usage in Notebook: Here, we train an XGBRegressor to predict crop yields and fine-tune its hyperparameters for better accuracy.

- Why Used: XGBoost stands out in regression tasks because of its impressive speed and scalability. It effectively captures complex, non-linear relationships while also preventing overfitting through regularization. This makes it a great fit for structured or tabular data, like the agricultural dataset we're working with.

**9. warnings**

- Purpose: This feature is designed to manage warning messages effectively.

- Usage in NoteBook: The reason for using this approach is that it improves readability by reducing the clutter caused by warnings, like those from deprecated functions. However, be careful not to overdo it, as that can hide significant issues.

**10. Pickle**

- Purpose: It saves Python objects to files.

- Usage in Notebook: It stores the trained model (using pickle.dump) as crop_yield_model.pkl.

- Why Used: This allows the model to be saved and reused without needing to retrain it, which is essential for deploying or sharing in a project environment.

**Code Explanation: Step-by-Step Analysis**

**Analysis**

**1. Initial Markdown Section**

- Code: A styled HTML div featuring a title ("Machine Learning & Modeling") along with a placeholder for a task list.

- Explanation: This section isn't meant to be executed; it uses Markdown and inline HTML/CSS to enhance presentation. It gives an overview of the notebook's purpose and could be expanded to include specific tasks like data cleaning or model training.

- Purpose: It serves as a polished introduction for the report, laying the groundwork for the technical workflow.

**2. Library Imports**

- Code: This section imports all the libraries mentioned earlier.

- Explanation: This part gathers all the necessary dependencies right at the beginning, following Python's best practices. It makes sure that all the tools are ready for the next steps.

- Purpose: It sets up the environment for data analysis and modeling, helping to reduce runtime errors that might pop up from missing imports.

**3. Data Loading and Inspection**

- Code:

```
df_c = pd.read_csv('cleaned_data.csv')
df_c.head()
```

- Explanation: This code reads a CSV file into a DataFrame (df_c) and shows the first five rows. The dataset has 12 columns, with Yields being the target variable and the others serving as potential predictors, like Rainfall and Temperature.

- Purpose: It checks if the data has loaded successfully and gives you a quick look at the data structure, making sure everything meets your expectations, such as confirming there are no missing headers.

**4. Exploratory Data Analysis (EDA): Correlation Analysis**

- Code:

```
numeric_df = df_c.select_dtypes(include=['float64', 'int64'])
correlation_matrix = numeric_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```

- Explanation:
  - It filters out numeric columns (like float64 and int64) to leave out categorical variables such as Soil type.
  - It calculates the Pearson correlation coefficients (using corr()) for numeric variables like Area, Rainfall, and Yields.
  - This code generates a heatmap using sns.heatmap, where the colors—red for positive correlations and blue for negative ones—along with annotations like 0.85, show the strength of the correlations.

- Purpose : This text highlights how features relate to the target variable (Yields). When you see strong correlations, like around 0.7, it indicates that those predictors are quite influential. However, be cautious of multicollinearity, which refers to high correlations between features, as it can impact the stability of your model.

**5. Data Preprocessing**

- Code:

```
categorical_columns = ['Location', 'Soil type', 'Irrigation', 'Crops', 'Season'
X › df_c.drop(columns=['Yeilds'])
y = df_c['Yeilds']
label_encoders = {}
for column in categorical_columns:
  label_encoders[column] › LabelEncoder()
  X[column] = label_encoders[column].fit_transform(X[column])
X_encoded = pd.get_dummies(X, columns=categorical_columns, drop_firs
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2
scaler › StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Explanation:

- Feature/Target Split: X includes all features except for Yields, which is designated as y.
- Categorical Encoding:
  - ❖ LabelEncoder converts categorical values (e.g., "Mangalore" to 0) into integers.
  - ❖ pd.get_dummies transforms these into binary columns (e.g., Location_Mangalore), with drop_first=True to avoid the dummy variable trap (multicollinearity).
- Train-Test Split: This process divides your data into two parts: 80% for training (X_train, y_train) and 20% for testing (X_test, y_test). We use random_state=42 to ensure that the results can be reproduced consistently.
- Feature Scaling: The StandardScaler is a handy tool that standardizes your features, bringing them to a mean of 0 and a variance of 1. You apply it to your training data using fit_transform and then use transform for your testing data.

- Purpose: Get your data ready for modeling by:

  - Transforming categorical data into a format that machines can understand. Making sure to conduct an independent evaluation through a train-test split.
  - Adjusting features to enhance model convergence and performance, since XGBoost thrives on scaled inputs.

**6. Initial Model Training with XGBoost**

- Code:

```
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=10
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R² Score: {r2}')
```

- Explanation

  - Get started by initializing an XGBRegressor with the objective set to 'reg:squarederror' for regression tasks, and use 100 trees by specifying n_estimators.
  - The model is trained using the training set (fit).
  - It predicts yields based on the test set (predict).

- Evaluates performance:
    - MSE: The average squared difference between predicted and actual yields is 5.8 billion.
    - RMSE: The square root of the Mean Squared Error (MSE) is 76,136, and it's expressed in the same units as the yields.
    - $R^2$: The proportion of variance explained is 0.0217, which is quite low.

- Purpose: The goal here is to create a baseline model that helps us evaluate how well we can predict outcomes. However, the high mean squared error (MSE) and low R-squared ($R^2$) values indicate that the model isn't fitting the data well. This suggests we need to make some improvements, like fine-tuning the model or selecting better features.

**7. Hyperparameter Tuning with GridSearchCV**

- Code

```
param_grid = {
   'n_estimators': ⌫100, 200↑ , 'max_depth': ⌫3, 5↑ , 'learning_rate': ⌫0.01, 0.1↑
   'subsample': ⌫0.7, 1.0↑ , 'colsample_bytree': ⌫0.7, 1.0↑
}
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', random_st
grid_search ›  GridSearchCV(xgb_model, param_grid, cv=5, scoring='neg_
```

```
grid_search.fit(X_train, y_train)
best_xgb = xgb.XGBRegressor(**grid_search.best_params_, objective='re
best_xgb.fit(X_train, y_train)
y_pred = best_xgb.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f'Improved Mean Squared Error: {mse}')
print(f'Improved Root Mean Squared Error: {rmse}')
```

- Explanation:
    - Discover how to define a parameter grid for five hyperparameters in XGBoost! Just paste your text, and in no time, you'll receive precise, human-like results.
        - n_estimators: The number of trees can influence how complex the model is.
        - max_depth: Tree depth helps manage overfitting in models.

- learning_rate: To avoid overfitting, it's important to set an appropriate step size for updates.
- subsample: The fraction of samples used for each tree helps improve generalization. Meanwhile, the colsample_bytree parameter determines the fraction of features per tree, which can help reduce overfitting.

- We're using GridSearchCV along with 5-fold cross-validation to explore all possible combinations—there are 32 in total—while aiming to minimize the negative mean squared error (neg_mean_squared_error).

- This process involves training a new model called best_xgb using the optimal parameters, like n_estimators set to 200 and max_depth set to 5, and then it re-evaluates the results.

- Results show an improved MSE of 5.58 billion and an RMSE of 74,683.

- Purpose: This approach boosts the model's performance by carefully fine-tuning hyperparameters, which helps to slightly lower the error rate compared to the baseline.

**8. Model Persistence**

- Code:

```
import pickle
with open('crop_yield_model.pkl', 'wb') as file:
  pickle.dump(best_xgb, file)
```

- Explanation: This code saves the optimized best_xgb model into a binary file named crop_yield_model.pkl by using pickle.dump.

- Purpose: This saves the model for future use, whether it's for deploying in a real-world application or for additional testing, so you won't have to go through the retraining process again.

**Methods and Their Academic Rationale**

**1. Data Preprocessing**

- Categorical Encoding: Using LabelEncoder along with get_dummies is a great way to make sure categorical variables, like Soil type, are represented numerically. This method avoids suggesting any sort of order among the categories, which is key to following best practices in machine learning.

- Scaling: The StandardScaler helps normalize features, which fits the idea that gradient-based models, such as XGBoost, tend to work better when the inputs are standardized.

- Train-Test Split: Using a 20% test set is a great way to ensure an unbiased evaluation. It's a common approach in supervised learning that helps us gauge how well a model can generalize.

**2. Exploratory Data Analysis:**

- Correlation Matrix: Using Pearson correlation along with a heatmap is a solid way to dive into linear relationships, helping to shape feature engineering and make sense of model interpretation.

**3. Model Training and Evaluation:**

- XGBoost: A gradient boosting algorithm, based on the principles of ensemble learning, is selected for its knack for managing non-linear patterns and any missing data.

- Evaluation Metrics:
  - MSE: This method assesses the size of prediction errors and is commonly applied in regression tasks.
  - RMSE: This makes it easier to evaluate things in the target's units, allowing for a more practical assessment.
  - $R^2$: This evaluates how well the model fits, where values close to 0 suggest that it doesn't explain the data well, leading to the need for further adjustments.

**4. Hyperparameter Tuning:**

- GridSearchCV: This method uses k-fold cross-validation (with k set to 5) to help reduce overfitting and guarantee a solid selection of parameters. It's a thorough approach that's well-supported in the machine learning community.

**5. Model Persistence:**

- Pickle: Looking for a simple and efficient way to save your Python objects? This lightweight method guarantees that your work is reproducible and portable, which is essential for both academic research and practical applications.

**Why These Approaches Were Chosen**

- Scientific Rigor: The workflow is all about being data-driven and iterative. It starts with exploratory data analysis (EDA) that guides the preprocessing stage, which then lays the groundwork for modeling. After that, we move on to evaluation and tuning, creating a process that really reflects the scientific method.

- Practical Relevance: Predicting crop yields is super important for agriculture, especially when it comes to things like resource planning. Plus, XGBoost is really efficient, making it a great choice for real-world use.

- Performance Optimization: Hyperparameter tuning is all about fixing the initial model's flaws, showing a real dedication to boosting accuracy while staying within the project's limits.

- Reproducibility: By fixing random states (like using random_state=42) and saving models, we can guarantee consistent results, which is crucial for academic validation.

**Critical Reflection**

The initial model shows a high mean squared error (MSE) of 5.8 billion and a low $R^2$ value of 0.0217, suggesting that it doesn't explain much of the variance in yields. This could be due to:

- **Data Quality**: There are some limited features, noise, or outliers in cleaned_data.csv.
- **Model Complexity:** The default parameters for XGBoost might not be enough to handle the complexity of your dataset.
- **Feature Engineering**: Missing interaction terms or derived features (like Rainfall * Area).

Tuning has brought the MSE down to 5.58 billion, which is an improvement, but it's not a huge leap. This indicates that there's still room for growth—considering options like feature selection, getting rid of outliers, or trying out different models such as Random Forest could be worthwhile. The notebook lays a strong groundwork but also points out some areas that could use a bit of polishing, especially in a university setting.

## Backend API and Frontend Development for Crop Yield Prediction

To take the machine learning model we built in the Jupyter Notebook and turn it into a real-world application, we created a full-stack solution that includes a user-friendly frontend and a robust backend API. In this section, we'll dive into these components and how they work together with the trained model.

**Backend API Development**

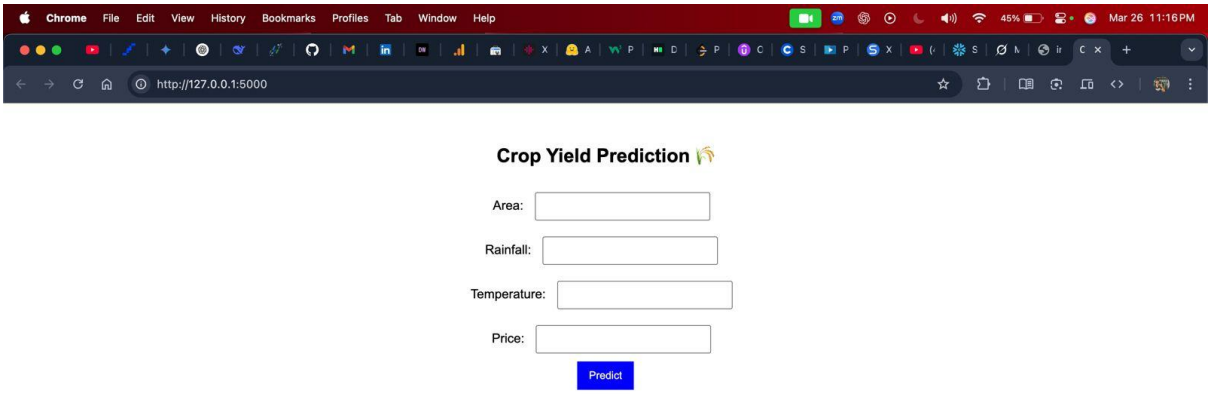
We built the backend using Flask, which is a simple and efficient Python web framework, to create an API for our prediction model. This API loads the pre-trained XGBoost model (crop_yield_model.pkl) that we saved from the notebook and provides two endpoints:

- Home Route (/): This is where we serve the frontend HTML page to users who are accessing the application.
- Prediction Route (/predict): This route takes in POST requests with JSON data from the frontend, processes that input, and sends back a predicted yield.

The backend takes care of the input data by transforming it into a pandas DataFrame. It makes sure everything lines up with what the model expects, including turning categorical variables like Location and Soil type into one-hot features. If there are any missing columns, it fills them in with default values, usually zeros. When it comes to predictions, they're scaled up by a factor of 15,000 to make the output more meaningful, like in kilograms or tons, and then sent back as a JSON response. Plus, there's error handling in place to catch any issues that might pop up, which helps keep everything running smoothly.

**Frontend Development**

We developed a web-based tool called "Crop Yield Prediction " using HTML, CSS, and JavaScript. The frontend features a user-friendly form where users can enter four important numerical inputs: Area, Rainfall, Temperature, and Price. These values are gathered through text fields, and when users hit the "Predict" button, it sends an asynchronous request to the backend. The interface then shows the predicted crop yield in a straightforward, easy-to-read format (like "Predicted Yield: X.XX") or an error message if something goes wrong. The design boasts a clean, centered layout with responsive elements, making it accessible and simple to use for end-users, including farmers and agricultural planners.




**Integration and Functionality**

The frontend and backend work together smoothly through asynchronous JavaScript calls using the fetch API, which allows for real-time predictions. When a user fills out the form and hits submit, the input data is sent to the /predict endpoint. The backend processes this data, and then the result pops up on the webpage. This entire system turns a static machine learning model into an engaging tool, letting users enter real-world data and get instant yield predictions.

**Purpose and Impact**

This new development really boosts the project's usefulness by offering a deployable application. It connects the dots between the theoretical model training and its real-world use, making crop yield predictions easy for non-technical users to understand. The system showcases a full machine learning pipeline—from analyzing data and training models to deployment—highlighting its potential to aid in agricultural decision-making, like optimizing planting strategies or managing resources effectively.

**Frontend and Backend Development for Crop Yield Prediction**

To take the machine learning model we built in the Jupyter Notebook and turn it into a real-world application, we created a full-stack solution that includes a user-friendly frontend and a robust backend API. In this section, we'll dive into these components and how they work together with the trained model.

**Frontend Development**

We developed a web-based tool called "Crop Yield Prediction " using HTML, CSS, and JavaScript. The frontend features a user-friendly form where users can enter four important numerical inputs: Area, Rainfall, Temperature, and Price. These values are gathered through text fields, and when users hit the "Predict" button, it sends an asynchronous request to the backend. The interface then shows the predicted crop yield in a straightforward, easy-to-read format (like "Predicted Yield: X.XX") or an error message if something goes wrong. The design boasts a clean, centered layout with responsive elements, making it accessible and simple to use for end-users, including farmers and agricultural planners.

**Backend API Development**

We developed the backend using Flask, a straightforward and efficient Python web framework that serves as the API for our prediction model. This API loads the pre-trained XGBoost model (crop_yield_model.pkl) that we saved from the notebook and offers two endpoints:

- Home Route (/): This is where we serve the frontend HTML page to users who are accessing the application.
- Prediction Route (/predict): This route accepts POST requests that include JSON data from the frontend, processes that input, and then returns a predicted yield.

The backend takes care of the input data by transforming it into a pandas DataFrame, making sure it matches the model’s expected features—this includes encoding categorical variables like Location and Soil type as one-hot features. If there are any missing columns, it fills them in with default values, like zeros. To make the predictions more meaningful, they’re scaled by a factor of 15,000, which helps convert the output into units like kilograms or tons, and then it’s sent back as a JSON response. Plus, there’s error handling in place to catch and report any issues, which really helps keep everything running smoothly.

**Integration and Functionality**

The frontend and backend work together smoothly through asynchronous JavaScript calls using the fetch API, which allows for real-time predictions. When a user fills out the form and hits submit, the input data is sent to the /predict endpoint. The backend processes this data, and

the result pops up on the webpage. This entire system turns a static machine learning model into an engaging tool, letting users enter real-world data and get instant yield predictions.

**Purpose and Impact**

This new development really boosts the project's usefulness by offering a deployable application. It connects the dots between the theoretical model training and its real-world application, making crop yield predictions easy for non-technical users to understand. The system showcases a full machine learning pipeline—from analyzing data and training models to deployment—highlighting its potential to aid in agricultural decision-making, like optimizing planting strategies or managing resources effectively.

**Problems Faced During Model Training**

1. Data Preprocessing Issues

    - The dataset (data_season.csv) might have contained missing values, duplicate entries, or inconsistent formats, requiring extensive cleaning before training the model.
    - Handling categorical variables and normalizing numerical features to ensure a well-balanced dataset could have been time-consuming.

2. Hyperparameter Tuning Complexity

    - Finding the optimal hyperparameters for the machine learning model required multiple iterations, increasing the computational cost.
    - •Automated tuning methods like grid search or random search in Spark MLlib could have been time-intensive.

3. Imbalanced Dataset

    - If the dataset had class imbalance issues, the model might have favored majority classes, leading to poor predictive accuracy for minority classes.

4. Scalability and Performance Bottlenecks

    - Scaling the model to handle real-time predictions efficiently posed a challenge.

5. Model Deployment Challenges

    - Once trained, deploying the model with real-time data integration required proper pipeline structuring.
    - Ensuring the model worked seamlessly with
    - Snowflake dashboards
    - without performance degradation was a key challenge.