**BDA Project Report**

**Project 4 (Agricultural Data)**

**Group members:**

Shahmeer Khan 25156

Muhammad Saim Khalid 25256

Muhammad Mashhood 24528

**Dataset URL**

<https://www.kaggle.com/datasets/noeyislearning/agricultural-production?select=agricultural_production_census_2007.csv>

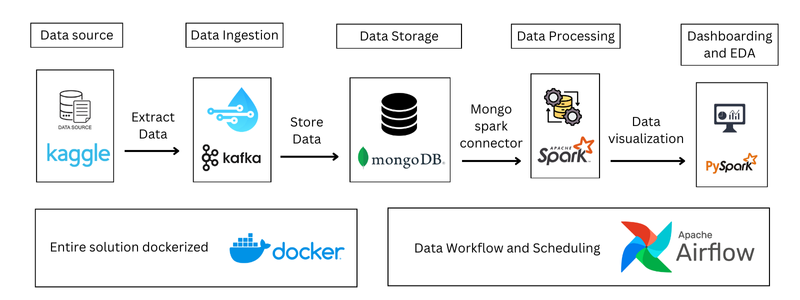
**Git URL**

https://github.com/shahmeerkm10/BDA\_solution

**Youtube video demo:**

<https://youtu.be/WK3et7hRJMc>

**BDA Architecture**

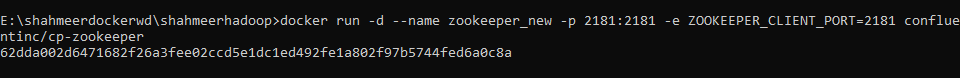
****

**BDA Implementation**

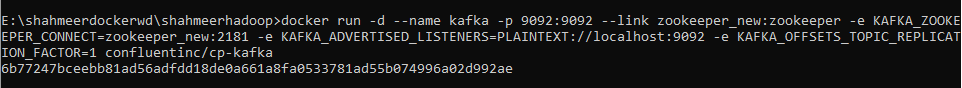
We started our project off by first ingesting the data via Kafka

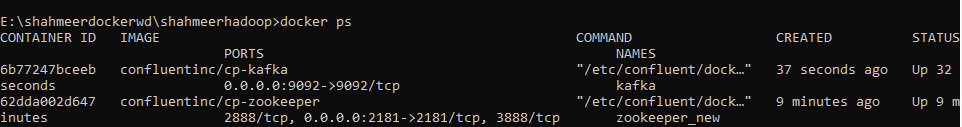
**- Kafka**

Started a new zookeeper instance for kafka integration

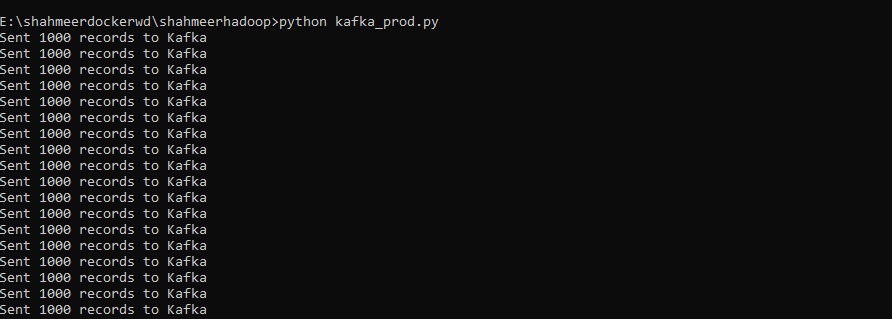
****

Initialized a Kafka container and topic for our data and linked it to our zookeeper instance

****

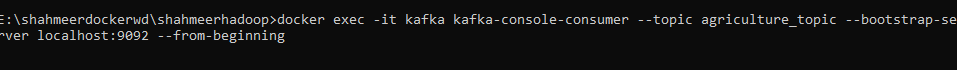
****

Executed a Kafka Producer Script to simply ingest the data into Kafka (included in attached zip file), 1000 records per command due to chunk size being 1000

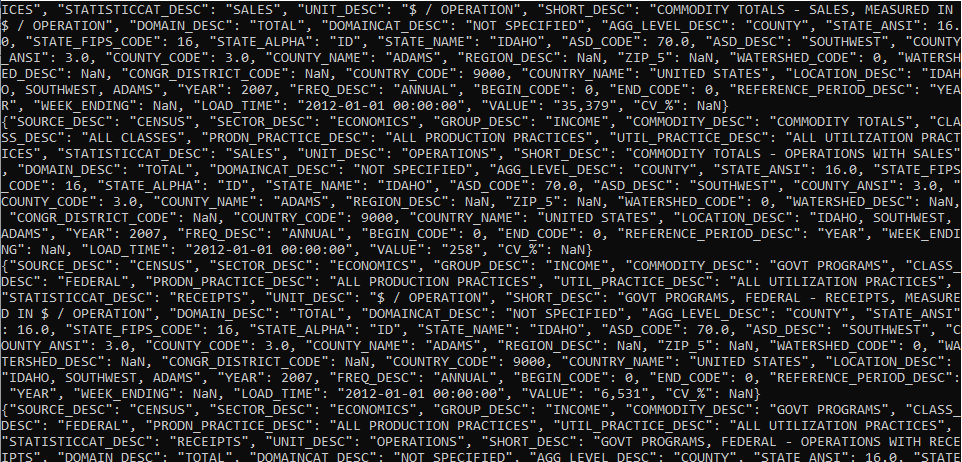
****

****

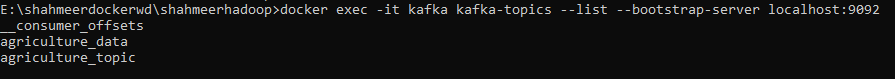
Running a Kafka consumer instance to check whether data is being written properly



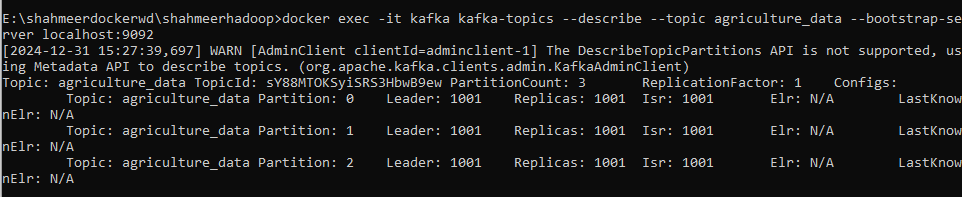
Kafka consumer status after running producer script

****

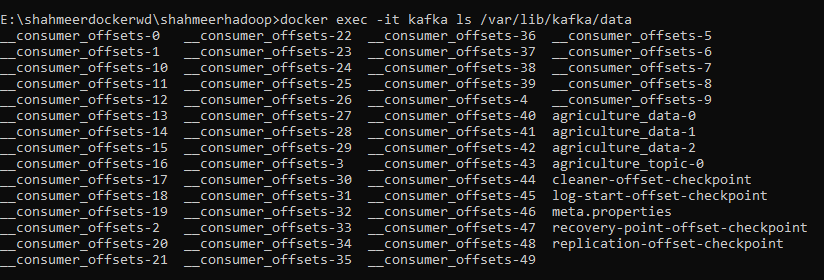
Details of the topics created

****

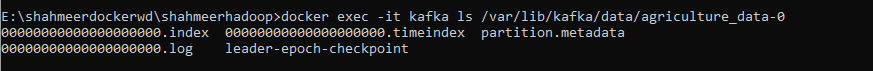
Details of the agriculture data topic which we used to process our data

****

Accessing the data folder of Kafka to see our agriculture data (3 partitions)

****

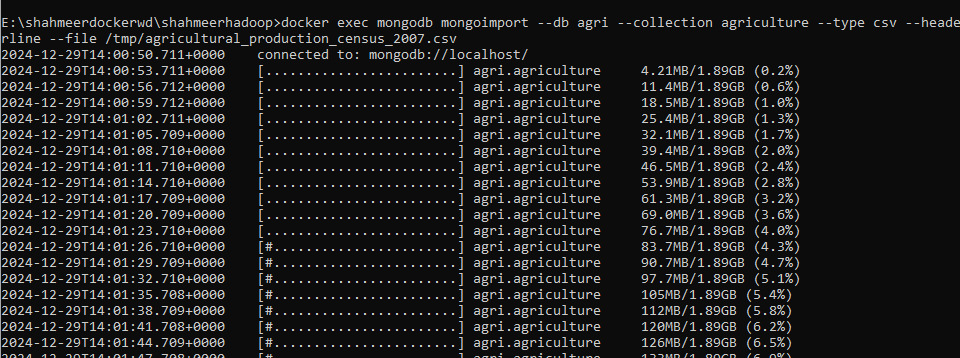
Checking one of those partitions to see log file (where data is stored)

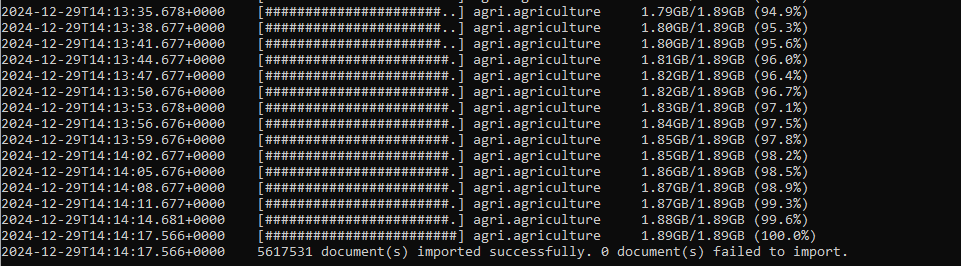
****

Kafka however kept crashing on attempts with different chunk sizes when importing directly into mongo container so had to resort to direct mongo import. System kept crashing after certain number of message writes or would get stuck in a loop.

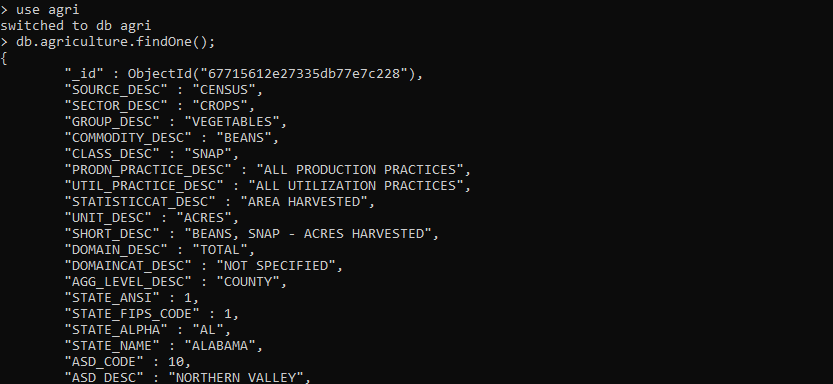
**Mongo**

We imported the dataset directly into mongo as Kafka to mongo became too heavy of a load for our systems



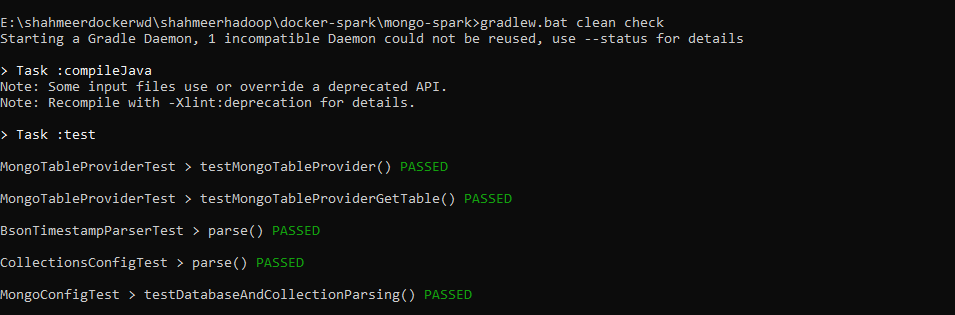


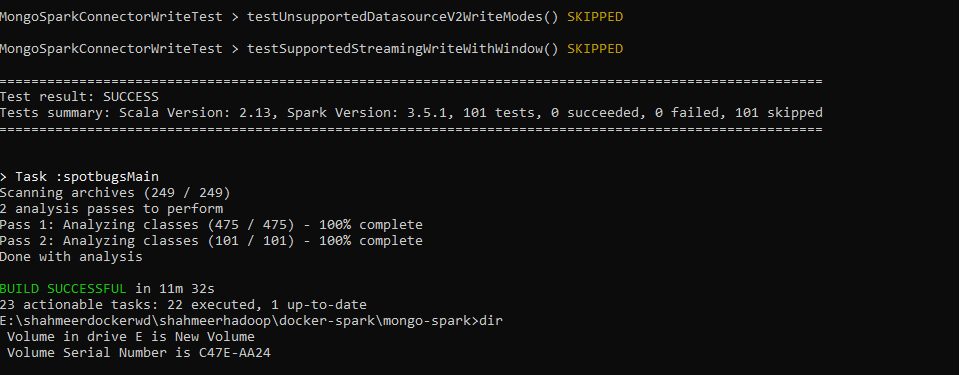
Proof of import success



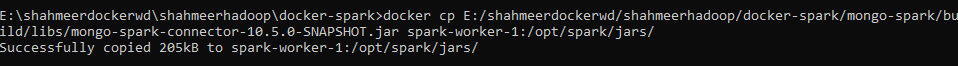
**Integrating spark with mongo**

Installing mongo spark (git pull) and building the connector



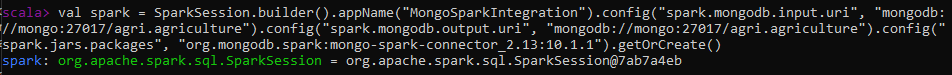


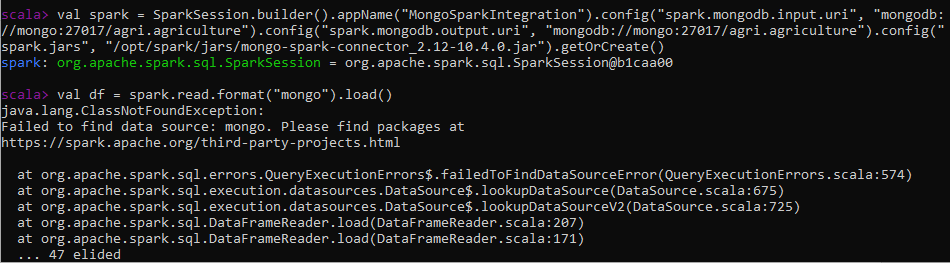
Copying the jar files into worker and master nodes

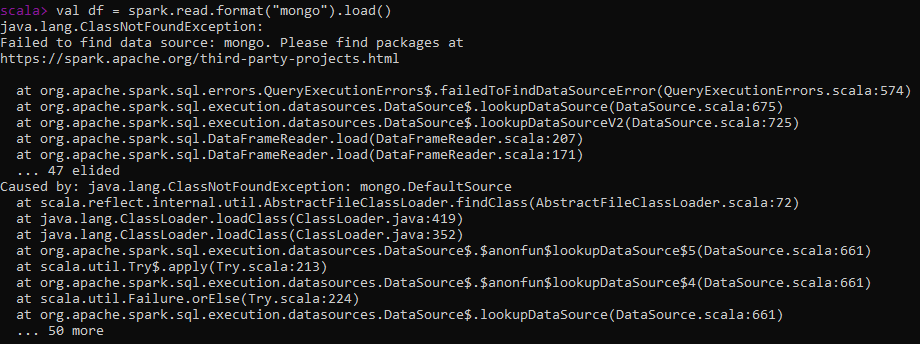


Gave errors after creating the spark session

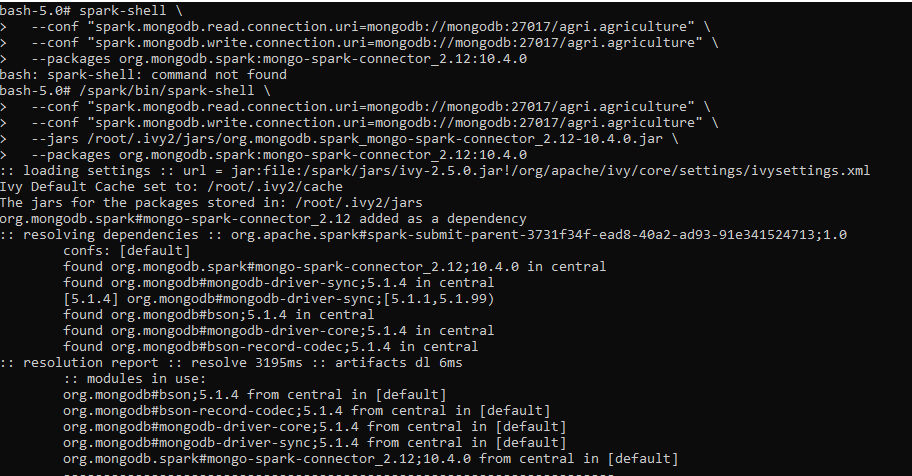
Attempted multiple different connector files but none seemed to work for some reason as the file had errors loading, used both “spark.jars.packages” and “spark.jars” to allow download and use local file



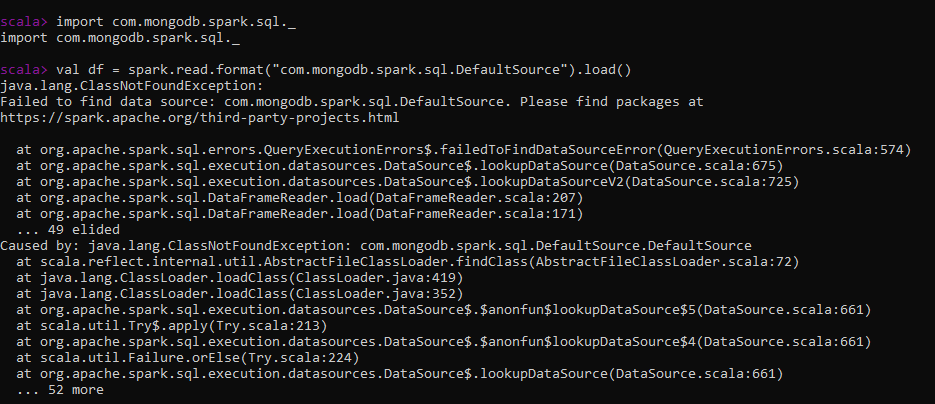




Also attempted by adding the configuration directly into shell



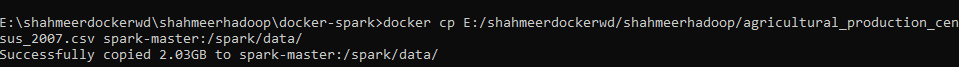
Still faced the same error



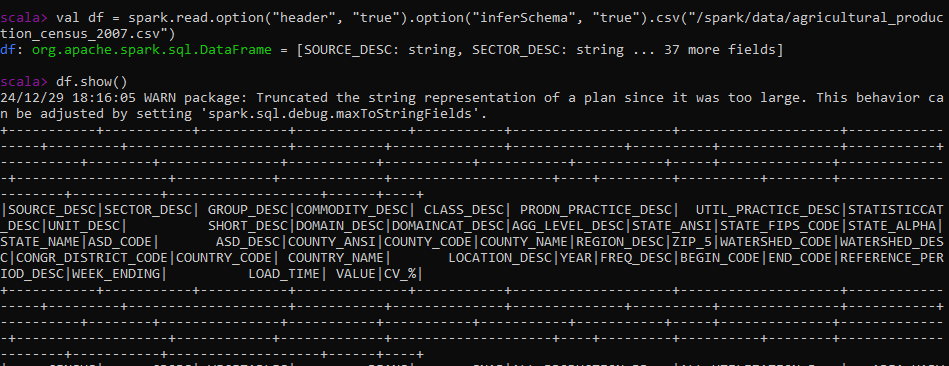
Due to mongo and spark not linking, had to import the csv file directly into spark to perform eda and transformations etc

**Spark**

Loading the data into spark

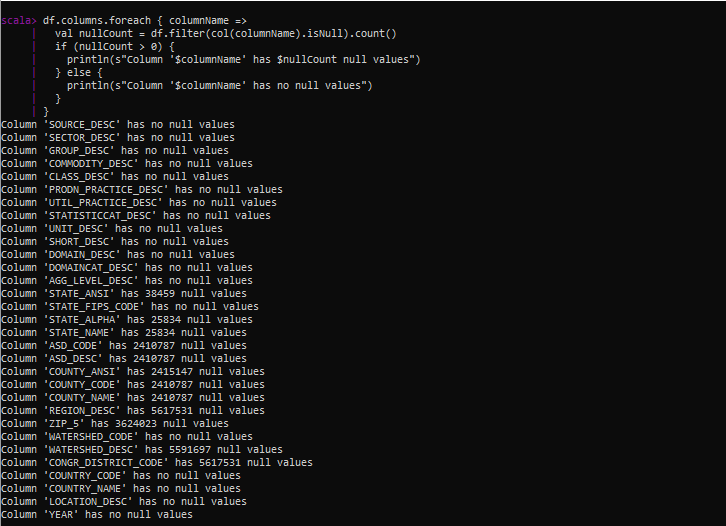


Proof of data being loaded successfully



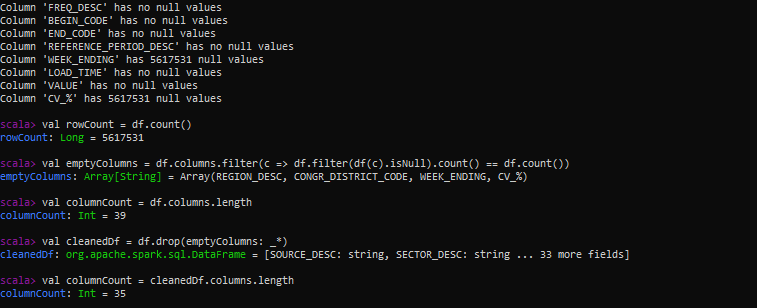
**Data preprocessing**

For starters wechecked column based null values to see how many nulls each column had

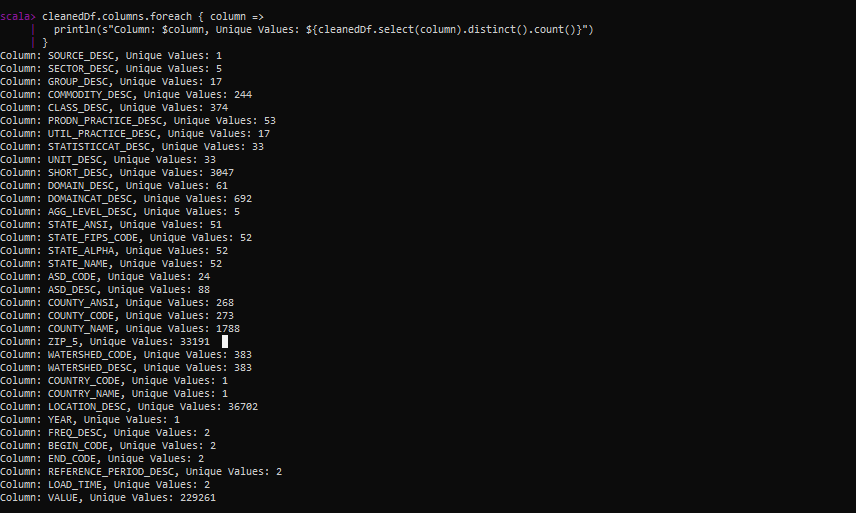


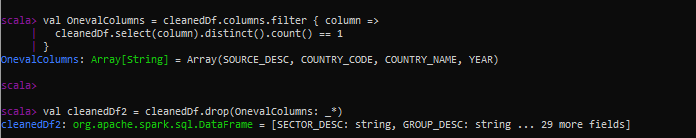
We then checked total no of rows in dataset and found a few columns that are fully empty

Named them and removed them



Identified and removed columns with a singular value as they add nothing to our data analysis but uses unnecessary storage space

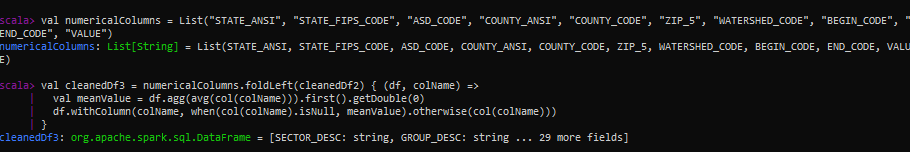






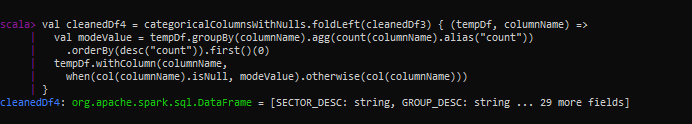
**Numeric missing vals**

Identified columns with missing values that are numeric and used col based mean values to handle them

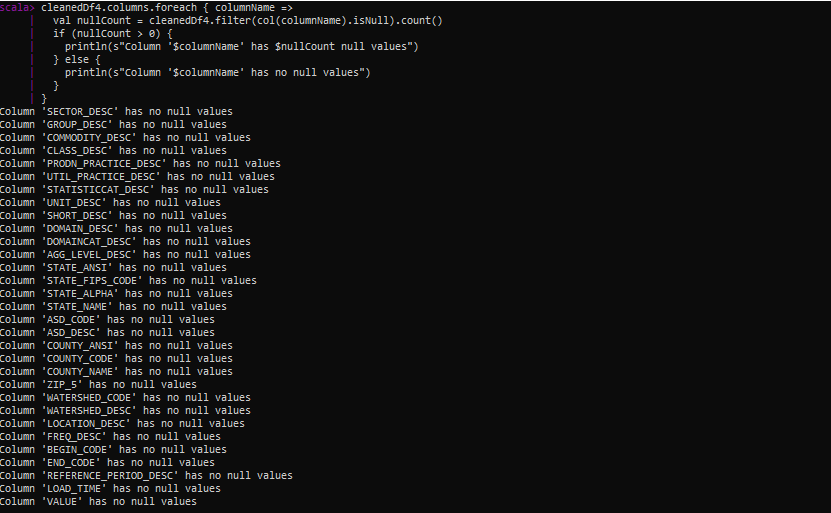


**Categorical missing vals**

Used modal values for imputation

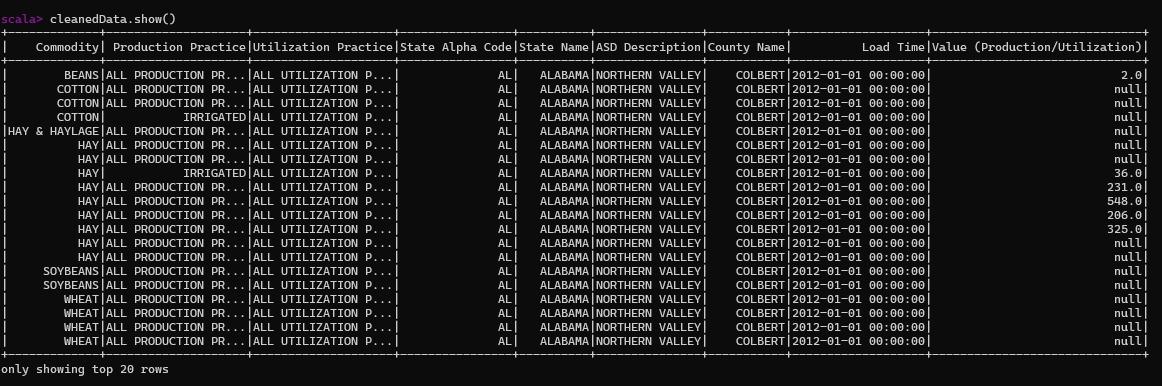


Updated dataframe with no more missing values left, we also renamed all columns for simplicity

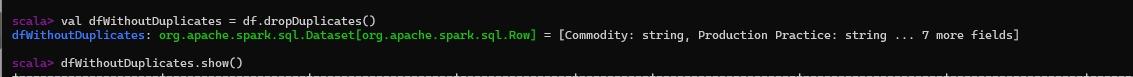


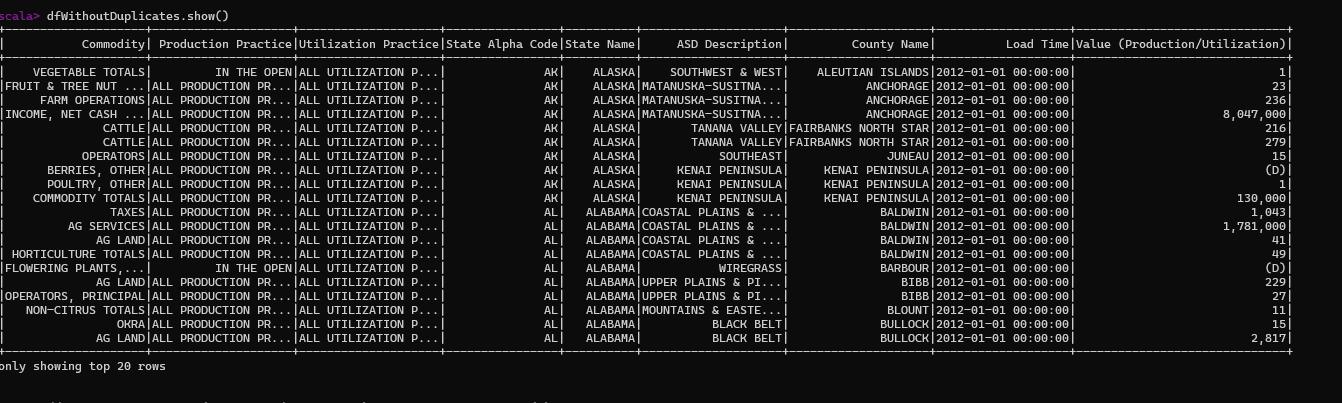
**Handling duplicate rows**

Before deletion



After deletion



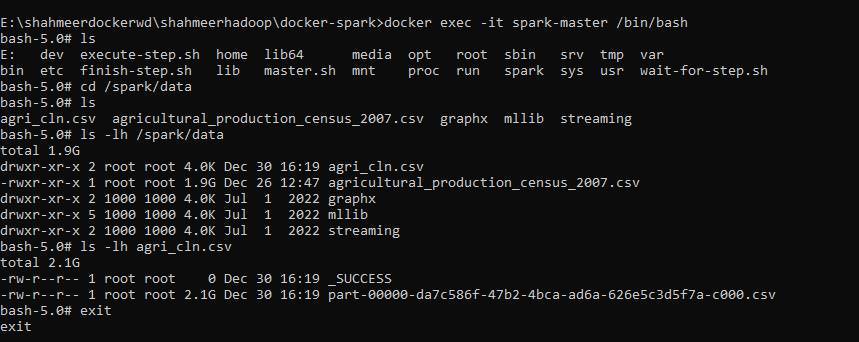


To then use this dataset for further processing on our other systems, wrote it into a csv file

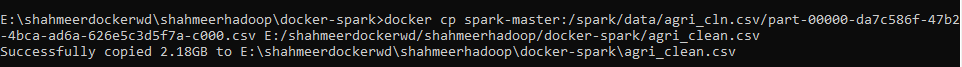
To copy this file into local storage, first write it into a csv into spark masters data folder, did this as copying into local system directly wasn’t possible due to dockerization, used coalesce to repartition the data into a single instance as spark would normally write it into several files in one folder



To get file name from spark master

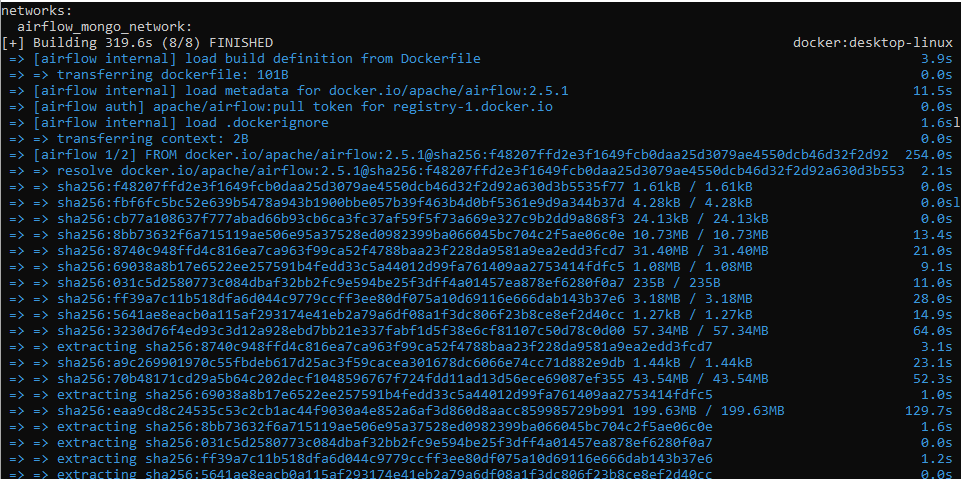


Copy into local storage

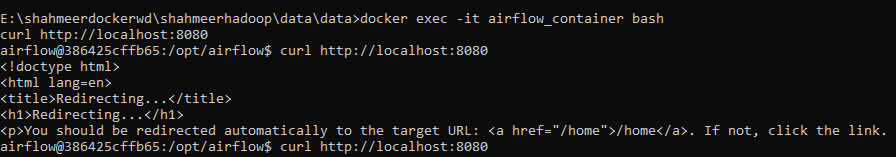


**Airflow**

Running the compose file with a few changes i.e. linking my existing mongodb container (edited compose file attached in zip)

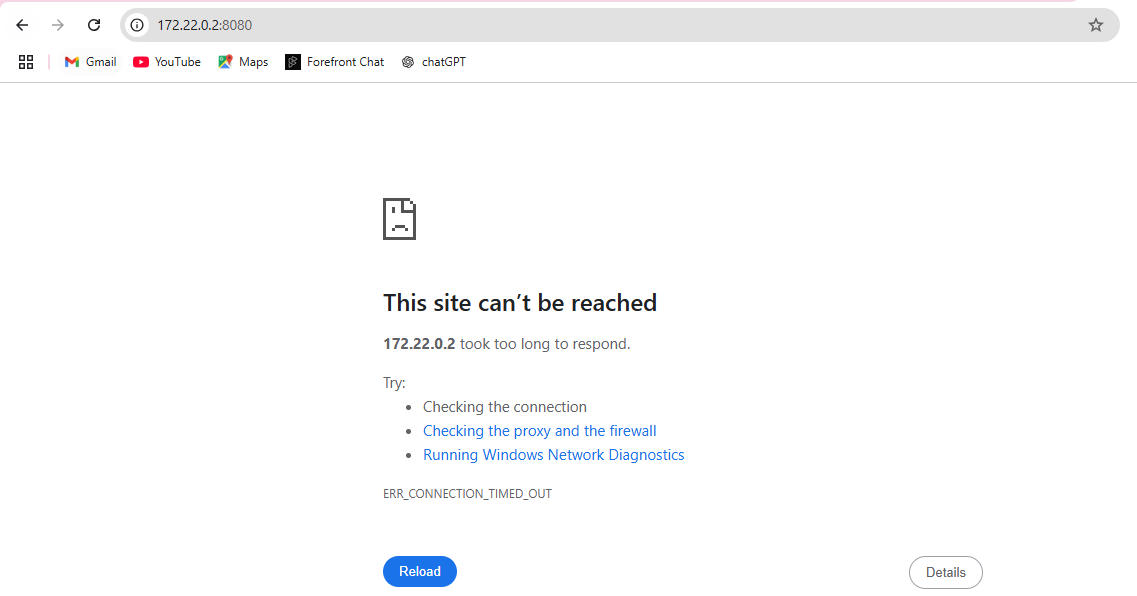


Bashing into airflow shell

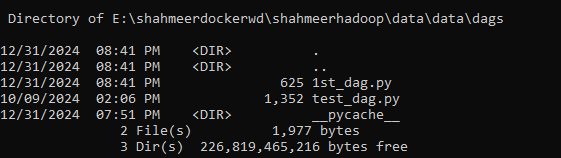


Localhost url (web ui) for some reason didn’t work

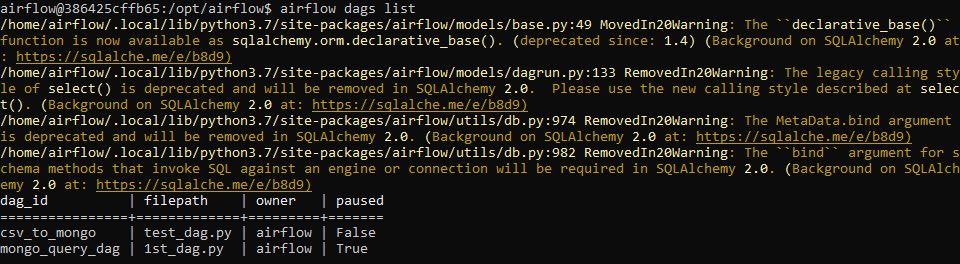
Allowed firewall access for both ports 8080 and 8082 but still got the same error so had to use cli directly



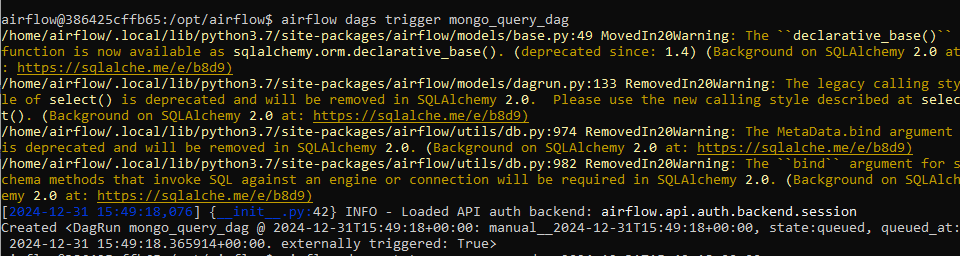
Created a sample dag “1st\_dag” linked to my mongo db (dag file attached in zip)



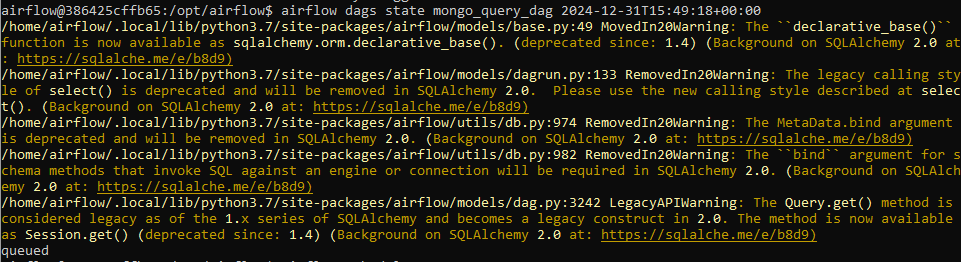
Airflow recognized the dag



Triggered the dag



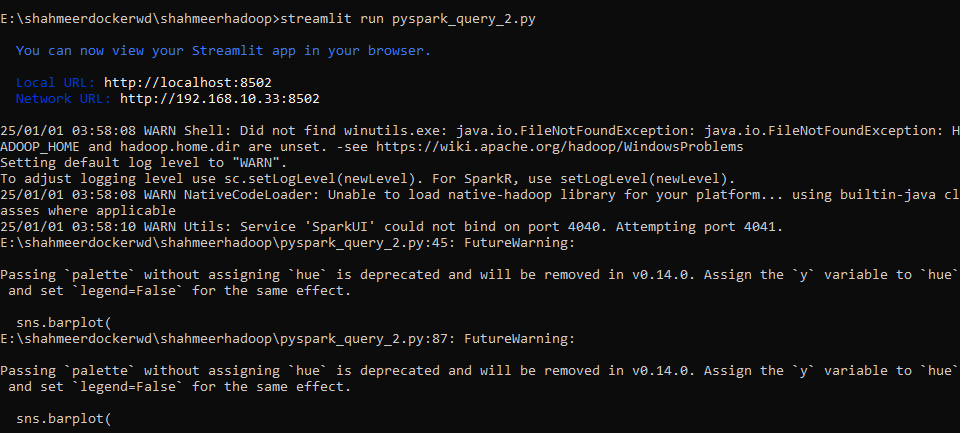
Dag status shown as queued, couldn’t resolve on why this was occurring, attempted restarting the scheduler as well as the webserver but that didn’t help



**BI Analysis**

For EDA and dashboarding we chose a set of BI queries based on our dataset, we renamed the columns for our ease of understanding.

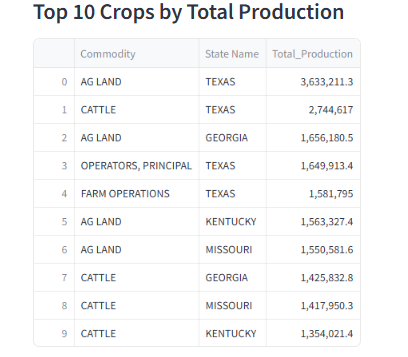
For BI analysis we used pyspark integrated with streamlit (streamlit python file added to zip)

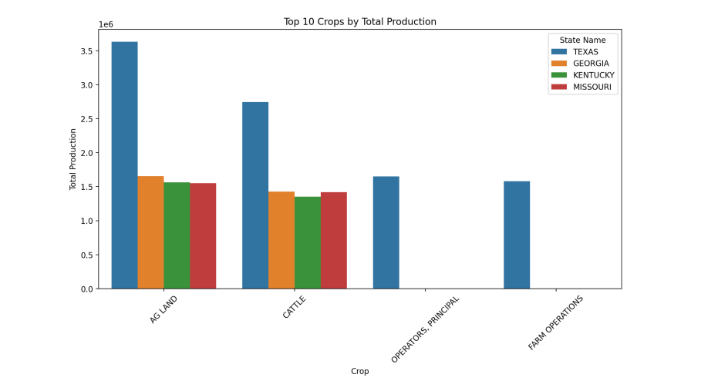


**Queries:**

* **Which crops had the highest production across different states and regions?**
* **How does agricultural production vary based on commodity and utilization practice across different regions?**
* **What is the relationship between the harvested crop area and the agricultural practices used?**
* **Which states and countries have the highest crop production and what are the trends over time?**
* **How do production statistics vary by year and region?**
* **Identify the underutilized agricultural resources and regions with growth potential**

**Query 1:**





#### 

#### 

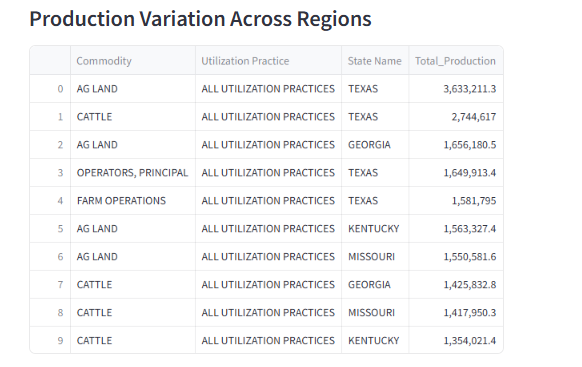
#### What it does:

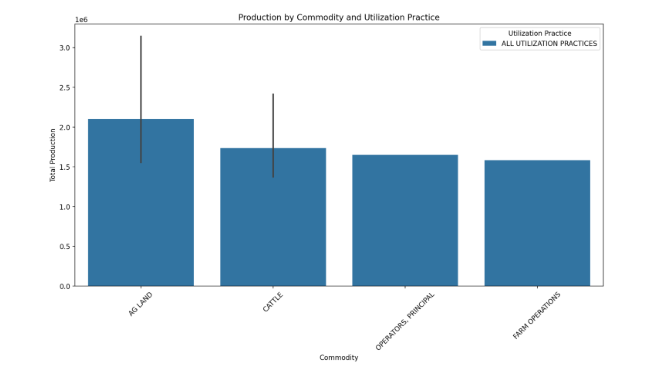
* Aggregates total production (Value (Production/Utilization)) for each crop (Commodity).
* Orders crops by total production in descending order.
* Displays the top 10 crops with the highest production using a bar chart.

#### Insights:

* Identifies the most productive crops across all regions.
* Highlights which crops are critical to overall agricultural output.
* Useful for stakeholders to focus on high-production crops for investment, research, or marketing

Query 2:





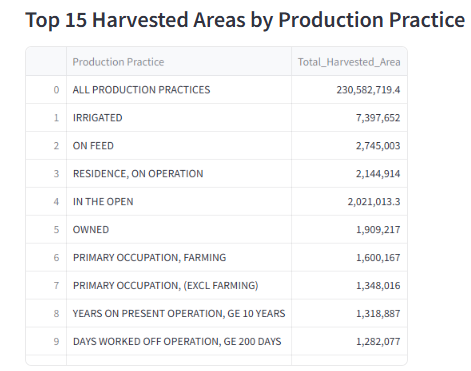
#### What it does:

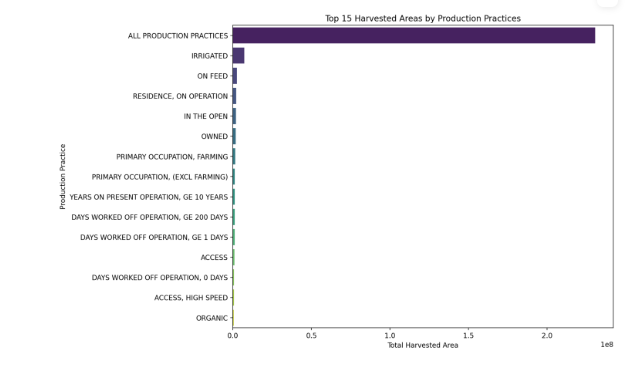
* Aggregates total production for each combination of crop (Commodity) and its **utilization practice** (e.g., food, feed, biofuel).
* Displays a **dotplot** to show how production is distributed across different commodities and their usage.

#### Insights:

* Provides an understanding of how crops are utilized (e.g., for human consumption vs. animal feed).
* Highlights commodities with diverse or niche utilization practices.
* Helps identify underutilized commodities with growth potential in alternative markets.

Query 3:





#### 

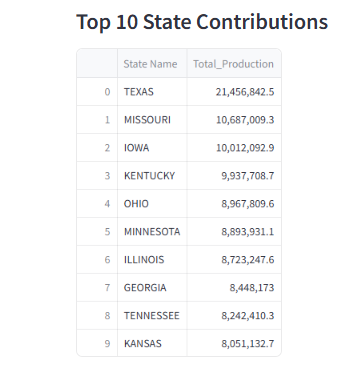
#### What it does:

* Aggregates the total harvested area for each **agricultural production practice** (e.g., organic, conventional farming).
* Orders the production practices by harvested area in descending order.
* Displays the **top 15 production practices** using a **horizontal bar chart**.

#### Insights:

* Identifies the most widely adopted farming practices based on harvested area.
* Highlights the importance of sustainable practices (e.g., organic farming) versus conventional practices.
* Helps policymakers and researchers understand which practices dominate and where resources can be directed for improvement.

Query 4:



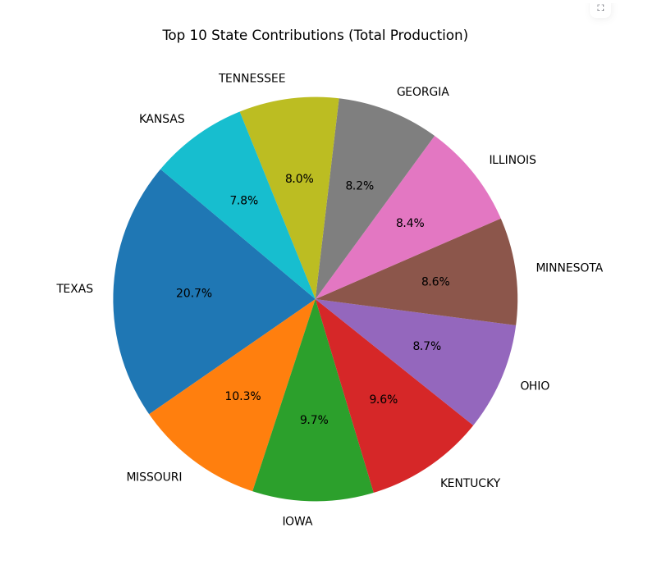
#### What it does:

* Aggregates total crop production for each **state**.
* Displays the **top 10 states** with the highest total production using a **pie chart**.

#### 

#### Insights:

* Shows the states contributing the most to agricultural production.
* Offers a proportional view of production across top-performing states.
* Helps identify regions that could act as benchmarks for agricultural policies and technologies.



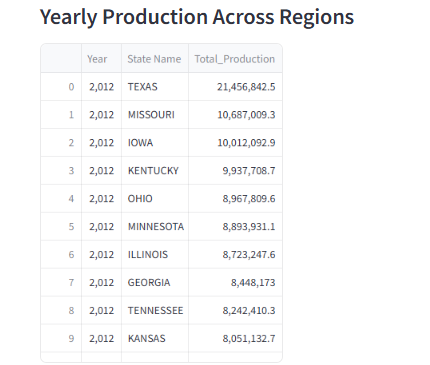
#### What it does:

* Aggregates total crop production for each **state**.
* Displays the **top 10 states** with the highest total production using a **pie chart**.

#### Insights:

* Shows the states contributing the most to agricultural production.
* Offers a proportional view of production across top-performing states.
* Helps identify regions that could act as benchmarks for agricultural policies and technologies.

**Query 5:**

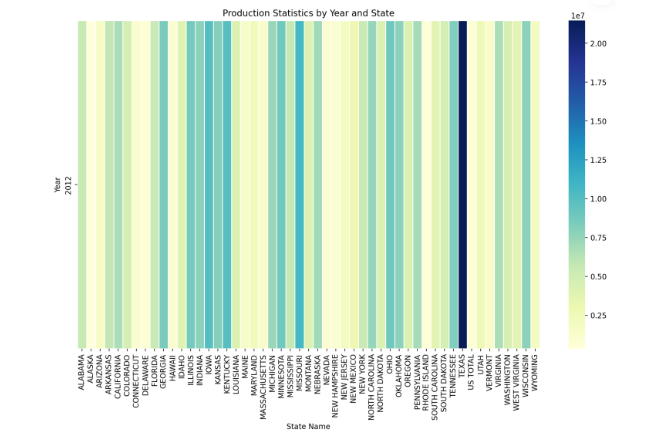


#### What it does:

* Aggregates total production by year and **region** (ASD Description).
* Displays a **line chart** to show trends in production across regions over time.

#### Insights:

* Identifies production trends over the years (e.g., growth, decline, or stagnation).
* Highlights regions with consistent or volatile production patterns.
* Useful for understanding the long-term impacts of policies, climate, or market demands on regional agriculture.



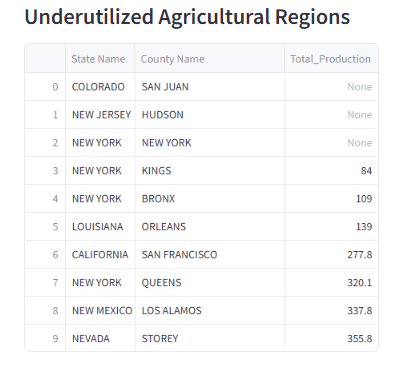
#### What it does:

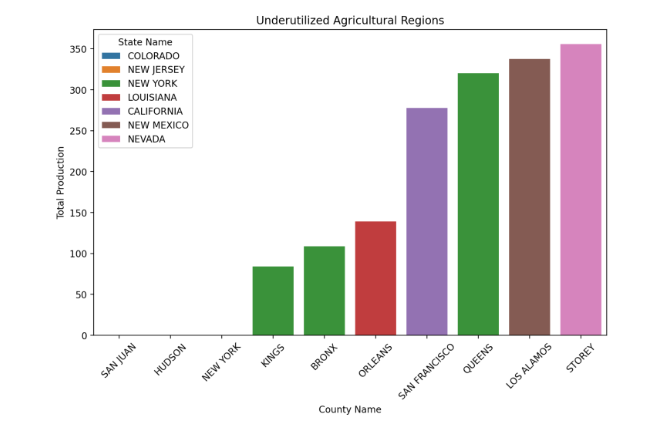
* Aggregates total production by year and **region** (ASD Description).
* Displays a **heat map** to show trends in production across regions over time.

#### Insights:

* Identifies production trends over the years (e.g., growth, decline, or stagnation).
* Highlights regions with consistent or volatile production patterns.
* Useful for understanding the long-term impacts of policies, climate, or market demands on regional agriculture.

**Query 6:**





#### 

#### What it does:

* Filters regions with **low production values** (below a threshold of 10,000).
* Groups data by counties and aggregates the total production.
* Displays the results using a **bar graph** to show the proportion of underutilization across counties.

#### Insights:

* Identifies regions with underutilized agricultural resources or low productivity.
* Highlights areas with growth potential for future agricultural development.
* Useful for directing investment, infrastructure, or research into underperforming regions.