**Project Report: Brewery Data Analysis**

Note: *This report outlines the steps taken during the analysis of brewery data, covering data cleaning, preliminary analysis in Hive, and more detailed processing and analysis using PySpark.*

1. Data Extraction and Initial Analysis

The dataset was obtained by downloading from the internet, and the extraction process involved using wget and unzip commands on the VM machine.

A screen shot of a computer

Description automatically generated

Wget to download the file directly from internet to the device.

A computer screen with white text

Description automatically generated

Unzip the downloaded file after renaming the same.

Following extraction, a Hive table was created, and the data was imported into the Hive database. Preliminary analysis revealed that the dataset was relatively clean, with the only adjustment made to the date field for proper formatting.

A screenshot of a computer

Description automatically generated  
  
Source : [Brewery Operations and Market Analysis Dataset (kaggle.com)](https://www.kaggle.com/datasets/ankurnapa/brewery-operations-and-market-analysis-dataset)

2. Preliminary Hive Query

A Hive query was executed to perform initial testing and data checking:

SELECT Beer\_Style, SKU, SUM(Total\_Sales) AS Total\_Sales

FROM brewery\_data

GROUP BY Beer\_Style, SKU

ORDER BY Total\_Sales DESC

LIMIT 10;

1. Top 10 Selling Beers:

A screenshot of a computer

Description automatically generated

2. Average Efficiency by Beer Style

A screenshot of a computer

Description automatically generated

3. Data Processing and Analysis with PySpark

A screenshot of a computer

Description automatically generated

3.1 Counting Rows:

A Spark operation to count rows was performed to ensure data availability and to check for potential errors.

A screenshot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated

3.2 Distribution of Losses:

A complex Spark query was executed to visualize the distribution of losses across batches and styles. Due to the large dataset, the analysis took considerable time.

A screenshot of a computer program

Description automatically generated

Due to the sheer volume and overall slower speeds to extract the data from the system. We Moved to Running the same locally.

Reference : [Installation — PySpark 3.5.0 documentation (apache.org)](https://spark.apache.org/docs/latest/api/python/getting_started/install.html#:~:text=After%20activating%20the%20environment%2C%20use%20the%20following%20command,%23%20can%20also%20add%20%22python%3D3.8%20some_package%20%5Betc.%5D%22%20here)

Installed spark onto an existing conda installation using.

Conda install pyspark  
and ran the commands on that to demonstrate the usage regardless of where the same is running from.

Referenced the functions section to run a few of the analysis on pyspark  
Reference link : [Functions — PySpark 3.5.0 documentation (apache.org)](https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/functions.html)

Spark running on VSCode Locally   
  
A screenshot of a computer program

Description automatically generated

Results   
A screenshot of a computer

Description automatically generated

# *Distribution of losses across batches and styles*A normal run based on the same.  
A screenshot of a computer program

Description automatically generated

4 Data Transformation

A new column representing the ratio of total sales to volume was created, showcasing Spark's ability for data transformation.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

5. Error Mitigation and Visualization Optimization

Several errors were encountered during the process, particularly when trying to use Pandas and Matplotlib for visualization. To address this, the dataset was limited to the first 1000 rows, and Seaborn was used as an alternative to Matplotlib for specific visualizations at some points. Screenshots were captured to document these errors.

When this command was run without consideration:

# Visualize the impact of temperature on bitterness

temp\_vs\_bitterness = df.groupby("Temperature").agg(F.mean("Bitterness").alias("Avg\_Bitterness")).toPandas()

temp\_vs\_bitterness.plot(kind="scatter", x="Temperature", y="Avg\_Bitterness", title="Temperature vs. Bitterness")

plt.show()  
  
It threw the whole thing into an error fest.

A screenshot of a computer program

Description automatically generated

During our data analysis using PySpark and Pandas, we encountered challenges related to the size of the dataset, leading to issues with Pandas and Matplotlib. (BDAT 1004) To ensure a smooth demonstration and avoid deep troubleshooting, we took strategic steps:

1. Limiting Dataset Size:

Given the large dataset, we decided to limit the number of entries for demonstration purposes. By selecting a subset of data, specifically the first 1000 rows, we aimed to expedite code execution and visualization rendering for efficient troubleshooting.

```python

# Sample the first 1000 rows from the DataFrame

sample\_df = df.limit(1000)

```

2. Professional Approach:

While these adjustments were made for practical reasons, it's important to note that the decisions were made strategically to meet the goals of the demonstration within the given time constraints. In a professional setting, thorough troubleshooting and optimization would be conducted to handle larger datasets seamlessly.

By implementing these measures, we achieved a balance between effective demonstration and addressing challenges posed by the dataset's scale. This pragmatic approach allowed us to focus on showcasing key functionalities and insights without compromising the integrity of the analysis.

Visualisation results

A screenshot of a computer

Description automatically generated

# *Visualize the correlation between quality score and total sales*A screen shot of a computer screen

Description automatically generated

# *Visualize the relationship between brewhouse efficiency and total volume produced*  
A screenshot of a computer screen

Description automatically generated

# *Distribution of Quality Scores*

A screen shot of a graph

Description automatically generated

*Relationship between Fermentation Time and Quality Score*

A screenshot of a computer screen

Description automatically generated

# Bitterness Distribution

A screenshot of a computer screen

Description automatically generated

Sales performance across different locations

A screenshot of a computer screen

Description automatically generated

6. Conclusion

The data analysis journey involved a mix of Hive and PySpark, showcasing both challenges and solutions. Screenshots were taken to provide visual documentation of specific steps and errors encountered. The combination of local installation and subset sampling allowed for effective troubleshooting and visualization, demonstrating the power and flexibility of PySpark in handling large datasets. Further exploration and optimization can be pursued in future analyses.