Master PySpark: From Zero to Big Data Hero!!

Here's a structured set of notes with code to cover changing data types, filtering data, and handling unique/distinct values in PySpark using the employee data:

---

1. Changing Data Types (Schema Transformation)

In PySpark, you can change the data type of a column using the cast() method. This is helpful when you need to convert data types for columns like Salary or Phone.

```
from pyspark.sql.functions import col

# Change the 'Salary' column from integer to double
df = df.withColumn("Salary", col("Salary").cast("double"))

# Convert 'Phone' column to string
df = df.withColumn("Phone", col("Phone").cast("string"))

df.printSchema()
```

2. Filtering Data

You can filter rows based on specific conditions. For instance, to filter employees with a salary greater than 50,000:

```
# Filter rows where Salary is greater than 50,000
filtered_df = df.filter(col("Salary") > 50000)
filtered_df.show()

# Filtering rows where Age is not null
filtered_df = df.filter(df["Age"].isNotNull())
filtered_df.show()
```

3. Multiple Filters (Chaining Conditions)

You can also apply multiple conditions using & or | (AND/OR) to filter data. For example, finding employees over 30 years old and in the IT department:

```
# Filter rows where Age > 30 and Department is 'IT'
filtered_df = df.filter((df["Age"] > 30) & (df["Department"] ==
"IT"))
filtered_df.show()
```

4. Filtering on Null or Non-Null Values

Filtering based on whether a column has NULL values or not is crucial for data cleaning:

```
# Filter rows where 'Address' is NULL
filtered_df = df.filter(df["Address"].isNull())
filtered_df.show()

# Filter rows where 'Email' is NOT NULL
filtered_df = df.filter(df["Email"].isNotNull())
filtered_df.show()
```

5. Handling Unique or Distinct Data

To get distinct rows or unique values from your dataset:

```
# Get distinct rows from the entire DataFrame
unique_df = df.distinct()
unique_df.show()

# Get distinct values from the 'Department' column
unique_departments_df = df.select("Department").distinct()
unique_departments_df.show()
```

To remove duplicates based on specific columns, such as Email or Phone, use dropDuplicates():

```
# Remove duplicates based on 'Email' column
unique_df = df.dropDuplicates(["Email"])
unique_df.show()

# Remove duplicates based on both 'Phone' and 'Email'
unique_df = df.dropDuplicates(["Phone", "Email"])
unique_df.show()
```

6. Counting Distinct Values

You can count distinct values in a particular column, or combinations of columns:

```
# Count distinct values in the 'Department' column
distinct_count_department =
df.select("Department").distinct().count()
print("Distinct Department Count:", distinct_count_department)

# Count distinct combinations of 'Department' and
'Performance_Rating'
distinct_combinations_count = df.select("Department",
"Performance_Rating").distinct().count()
print("Distinct Department and Performance Rating Combinations:",
distinct_combinations_count)
```

This set of operations will help you efficiently manage and transform your data in PySpark, ensuring data integrity and accuracy for your analysis!

Mastering PySpark DataFrame Operations

1. **Changing Data Types**: Easily modify column types using .cast(). E.g., change 'Salary' to double or 'Phone' to string for better data handling.
2. **Filtering Data**: Use .filter() or .where() to extract specific rows. For example, filter employees with a salary over 50,000 or non-null Age.
3. **Multiple Conditions**: Chain filters with & and | to apply complex conditions, such as finding employees over 30 in the IT department.
4. **Handling NULLs**: Use .isNull() and .isNotNull() to filter rows with missing or available values, such as missing addresses or valid emails.
5. **Unique/Distinct Values**: Use .distinct() to get unique rows or distinct values in a column. Remove duplicates based on specific fields like Email or Phone using .dropDuplicates().
6. **Count Distinct Values**: Count distinct values in one or multiple columns to analyze data diversity, such as counting unique departments or combinations of Department and Performance_Rating.