|  |  |
| --- | --- |
| Pandas | Pyspark |
| 1. Read dataset   df = pd.read\_csv(‘Pharma.csv’) | df = spark.read.csv(os.path.join(path, 'pharma-data.csv'), header=True, inferSchema=True) |
| 1. Show first 10 records   df.head(10)  by default it show() displays only 5 records | df.show(10)  df.take(10)  by default show() displays only 20 records |
| 1. Show the info of the dataset   df.info() | df.printSchema() |
| 1. Describe a dataset   df.describe() | df.describe().show() |
| 1. Count the no. of rows and columns in a dataset   df.shape – gives both rows and columns  df.columns – give no. of columns | df.count() – give rows  len(df.columns) – give columns |
| 1. Get the data type of each column   df.dtypes | df.dtypes |
| 1. Missing values   df.isNull().sum() | from pyspark.sql.functions import col, sum  df.select([sum(col(c).isNull().cast("int")).alias(c) for c in df.columns]).show() |
| 1. Check duplicates   df.duplicated() | duplicate\_rows = df.groupBy(df.columns).count().filter("count > 1") |
| 1. Get distinct values in a column   df[‘Sales’].unique() | df.select(‘Sales’).distinct().show() |
| 1. Count and transform invalid date   df['new\_date'] = pd.to\_datetime(df['date'], errors='coerce', format='%Y-%m-%d') invalid\_date\_count = df[new\_date'].isna().sum() | Example date are of form ->  [("2023-10-01",), ("10/02/2023",), ("03-04-2023",), ("2023/10/05",), ("invalid\_date",)]  df\_with\_date = df.withColumn("standardized\_date", to\_date(col("date"), "yyyy-MM-dd")) # Count the number of invalid date formats (where the date could not be converted) invalid\_date\_count = df\_with\_date.filter(col("standardized\_date").isNull()).count() |
|  |  |

df.show(**truncate=False**) -> If True (default), it truncates columns to a default width of 20 characters. If False, it shows the full content of each column without truncating.

**errors='coerce'**: Tells Pandas to convert invalid or unprocessable data to **missing values** (NaT for datetime, NaN for numeric) rather than throwing an error.