
Nitya CloudTech
Dream.Achieve.Succeed
*PYSPARK
*PANDAS

Nitya CloudTech
Dream.Achieve.Succeed

# Why should we choose PySpark over Pandas when handling big data?


Nitya CloudTech
Dream.Achieve.Succeed

Imagine you have a giant notebook with millions of pages (Big Data).

- Pandas is like using a small desk to read and write in your notebook—it's fast but only works well if the notebook is small enough to fit on your desk.

- PySpark is like having hundreds of friends (Databricks Cluster) who help you read and write pages at the same time—so even if the notebook is HUGE, you can still finish fast!

pandas

APACHE Spark™


# 1. Scalability (Growing with More Work)

- **PySpark: You have a team of builders working together. Each person builds a part, so the castle gets built fast.**

- **Pandas: You work alone , so if the castle is too big, you run out of space on your table.**



## 2. Performance (How Fast It Works)

- **PySpark: Your team builds multiple walls at the same time —faster!**
- **Pandas: You build one wall at a time —slower for big projects.**


# 3. Big Data Support (Handling Huge Projects)

- **PySpark: You can build castles as big as a city because your team never runs out of space!**

- **Pandas: If the castle gets too big, your desk overflows, and you can't continue .**


# 4. Cluster Integration (Working in a Group)

- **PySpark: Your team talks to each other and shares work .**
- **Pandas: You work alone, with no helpers .**

pandas

APACHE Spark™


## 5. Fault Tolerance (What Happens If a Problem Occurs?)

- **PySpark: If one builder drops their part, someone picks it up and continues.**
- **Pandas: If you drop a piece, the whole castle might fall apart**


Nitya CloudTech
Dream.Achieve.Succeed

# PySpark DataFrame Methods (EDA & Data Manipulation)

## Creating DataFrames

### Creates a DataFrame from a list.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("EDA Methods").getOrCreate()
data = [("Alice", 25), ("Bob", 30)]
df = spark.createDataFrame(data, ["Name", "Age"])
df.show()
```

### Reads a CSV file into a DataFrame.

```
df = spark.read.csv("path/to/file.csv", header=True, inferSchema=True)
```

### Reads a Parquet file.

```
df = spark.read.parquet("path/to/file.parquet")
```

Reads JSON data.

```
df = spark.read.json("path/to/file.json")
```


# PySpark DataFrame Methods (EDA & Data Manipulation)

## Basic EDA Methods

Displays the first few rows of the DataFrame.

```
df.show(5)
```

Displays column names.

```
df.columns
```

Shows the number of rows.

```
df.count()
```

Displays the schema (column names and data types).

```
df.printSchema()
```

Computes summary statistics for numerical columns.

```
df.describe().show()
```

Computes column-wise missing values.

```
from pyspark.sql.functions import col, sum
df.select([sum(col(c).isNull().cast("int")).alias(c) for c in df.columns]).show()
```

Nitya CloudTech
Dream.Achieve.Succeed

# PySpark DataFrame Methods (EDA & Data Manipulation)

## Filtering & Selecting Data

Filters rows where Age is greater than 25.

```
df.filter(df.Age > 25).show()
```

Filters using multiple conditions.

```
df.filter((df.Age > 25) & (df.Name == "Alice")).show()
```

Selects specific columns.

```
df.select("Name", "Age").show()
```

Adds a new column.

```
from pyspark.sql.functions import lit
df = df.withColumn("Country", lit("USA"))
df.show()
```

Renames a column.

```
df = df.withColumnRenamed("Age", "Years")
```

Drops a column.

```
df = df.drop("Age")
```


# PySpark DataFrame Methods (EDA & Data Manipulation)

## Aggregations & Grouping

**Counts the number of occurrences per category.**

```
df.groupBy("Country").count().show()
```

**Finds the average age per country.**

```
df.groupBy("Country").agg({"Age": "avg"}).show()
```

**Computes multiple aggregations.**

```
from pyspark.sql.functions import avg, min, max
df.groupBy("Country").agg(avg("Age"), min("Age"), max("Age")).show()
```


# PySpark DataFrame Methods (EDA & Data Manipulation)

## Sorting & Ranking

**Sorts DataFrame in ascending order.**

```
df.orderBy("Age").show()
```

**Sorts in descending order.**

```
df.orderBy(df.Age.desc()).show()
```

**Adds a row number column (ranking).**

```
from pyspark.sql.window import Window
from pyspark.sql.functions import
row_number
window_spec =
Window.partitionBy("Country").orderBy("Age")
df.withColumn("RowNum",
row_number().over(window_spec)).show()
```


# PySpark DataFrame Methods (EDA & Data Manipulation)

## Handling Missing Data

**Drops rows with any null values.**

```
df.na.drop().show()
```

**Fills null values with a default value.**

```
df.na.fill({"Age": 30, "Name": "Unknown"}).show()
```

**Replaces specific values.**

```
df.replace("Alice", "Alicia").show()
```


# PySpark SQL Functions for Feature Engineering

## Conditional Logic

**Assigns categories based on age.**

```
from pyspark.sql.functions import when

df.withColumn("Category", when(df.Age < 18, "Child").otherwise("Adult")).show()
```

Nitya CloudTech
Dream.Achieve.Succeed

# PySpark SQL Functions for Feature Engineering

## String Functions

### Concatenates first and last names.

```
from pyspark.sql.functions import concat, lit
df.withColumn("FullName", concat(df.Name, lit(" Smith"))).show()
```

### Converts text to lowercase.

```
from pyspark.sql.functions import lower
df.withColumn("Name", lower(df.Name)).show()
```

### Removes whitespace from strings.

```
from pyspark.sql.functions import trim
df.withColumn("Name", trim(df.Name)).show()
```


# PySpark SQL Functions for Feature Engineering

## Mathematical Functions

### Rounds a number.

```
from pyspark.sql.functions import round
df.withColumn("RoundedAge",
round(df.Age, 0)).show()
```

### Computes the square root.

```
from pyspark.sql.functions import sqrt
df.withColumn("AgeRoot",
sqrt(df.Age)).show()
```

### Computes logarithm.

```
from pyspark.sql.functions import log
df.withColumn("LogAge",
log(df.Age)).show()
```


# PySpark SQL Functions for Feature Engineering

## Date & Time Functions

### Gets the current date.

```
from pyspark.sql.functions import
current_date
df.withColumn("Today",
current_date()).show()
```

### Gets the current timestamp.

```
from pyspark.sql.functions import
current_timestamp
df.withColumn("Now",
current_timestamp()).show()
```

### Extracts the year from a date column.

```
from pyspark.sql.functions import year
df.withColumn("Year",
year(df.DeliveryDate)).show()
```


# PySpark SQL Functions for Feature Engineering

## Vector Assembler (Feature Engineering)

```
from pyspark.ml.feature import
VectorAssembler
assembler = VectorAssembler(inputCols=
["Age"], outputCol="features")
df = assembler.transform(df)
df.show()
```

## Linear Regression

```
from pyspark.ml.regression import
LinearRegression
lr =
LinearRegression(featuresCol="features",
labelCol="Price")
model = lr.fit(df)
```


# PySpark SQL Functions for Feature Engineering

## K-Means Clustering

```
from pyspark.ml.clustering import KMeans
kmeans = KMeans(k=3, seed=1)
model = kmeans.fit(df)
```

## Pipeline for ML Modeling

```
from pyspark.ml import Pipeline
pipeline = Pipeline(stages=[assembler, lr])
model = pipeline.fit(df)
```


# PySpark SQL Functions for Feature Engineering

## Cache & Persist

Caches DataFrame in memory for faster access.

```
df.cache()
```

Persists DataFrame to memory/disk.

```
from pyspark import StorageLevel
df.persist(StorageLevel.MEMORY_AND_DISK)
```

Unpersists the DataFrame.

```
df.unpersist()
```

Nitya CloudTech
Dream.Achieve.Succeed

# PySpark SQL Functions for Feature Engineering

## Broadcast Joins (Optimizing Joins)

```
from pyspark.sql.functions import broadcast
df_large.join(broadcast(df_small), "ID").show()
```

## Repartitioning & Coalescing

Increases the number of partitions.

```
df.repartition(10)
```

Reduces partitions (better for small datasets).

```
df.coalesce(2)
```

## Explaining Query Execution

Displays execution plan for query optimization.

```
df.explain(True)
```


# PySpark Streaming Methods

## Reading Streaming Data

```
from pyspark.sql import SparkSession
spark =
SparkSession.builder.appName("Streamin
gExample").getOrCreate()
df =
spark.readStream.format("socket").option("
host", "localhost").option("port",
9999).load()
```

## Writing Streaming Data to Console

```
df.writeStream.outputMode("append").for
mat("console").start().awaitTermination()
```

Nitya CloudTech
Dream.Achieve.Succeed

# PySpark RDD Methods

## Creating an RDD

```
rdd = spark.sparkContext.parallelize([1, 2, 3, 4, 5])
```

## RDD Transformations

Doubles each element in the RDD.

```
rdd.map(lambda x: x * 2).collect()
```

Filters elements greater than 2.

```
rdd.filter(lambda x: x > 2).collect()
```

## RDD Actions

Counts elements.

```
rdd.count()
```

Finds the maximum value.

```
rdd.max()
```

Collects all elements.

```
rdd.collect()
```

Nitya CloudTech

Dream.Achieve.Succeed

# FOR CAREER GUIDANCE, CHECK OUT OUR PAGE

**www.nityacloudtech.com**

**Follow Us on Linkedin:**

**Aditya Chandak**

# Free SQL Interview Questions

Digital Product | 92 Sales

Helpful | Pyspark | Practical

Aditya Chandak offers valuable and practical insights, particularly in Pyspark and Data Engineering, helping greatly with interview preparation.

AI-generated based on testimonials

Are you preparing for SQL interviews? Don't miss this **FREE collection of SQL Interview Questions**, carefully curated to cover **real-world scenarios, advanced concepts, and tricky queries**.

🎯 **What's Inside?**

- ✅ Questions for beginners to advanced professionals.
- ✅ Scenario-based problems to test your skills.
- ✅ Focus on SQL optimization, joins, and query building.

💡 **Perfect for candidates aiming for top tech roles!**

Grab it now and give yourself the edge in your next SQL interview!

## Don't take it from me

Hear what others have to say

Very helpful

**Reyansh Srivastava**
Dec 2024

I high
knowled

Nitya CloudTech
Dream.Achieve.Succeed

# Free SQL Interview Preparation:

https://topmate.io/nitya_cloudtech/1403841

# Data Analyst Certification:

https://nityacloudtech.com/pages/courses/NCT_Courses

# Data Engineer Certification:

https://nityacloudtech.com/pages/courses/NCT_Courses

# Artificial Intelligence Certification:

https://nityacloudtech.com/pages/courses/NCT_Courses

# Register for Free AI Workshop:

https://nityacloudtech.com/pages/placement_training/AI_MLMasterClass

Nitya CloudTech

Dream.Achieve.Succeed