PySpark Full Practice Notes

Day1.py

```
# Databricks notebook source
from pyspark.sql import SparkSession
spark = SparkSession.builder\
    .appName("Py_Spark_Tutorial") \
    .getOrCreate()

print("Spark_successfully")

# COMMAND ----------

data = [
    (1,'A','Delhi',87654),
    (2,'B','Maharashtra',65547),
    (3,'C','Rajasthan',43523),
    (4,'D','Uttar Pradesh',43565),
    (5,'E','Haryana',67674),
]
columns = ['id','name','state','salary']
df = spark.createDataFrame(data,columns)
df.show()

# COMMAND ----------

# select
df.select("name","salary").show()

# COMMAND ----------

# filter
df.filter(df.salary > 50000).show()

# COMMAND ----------

# add columns
from pyspark.sql.functions import col
df2 = df.withColumn("salary_bonus",
                    col("salary")+5000)
df2.show()

# COMMAND ----------

#Task1
#Filter karo employees jo Delhi se hain
df2.filter(df.state == 'Delhi').show()

# COMMAND ----------

#Task2
#Salary > 48000 wale employees show karo
df2.filter(df2.salary > 48000).show()

# COMMAND ----------

#Task3
#Ek new column add karo: salary_after_tax = salary - 2000
from pyspark.sql.functions import col
df3 = df.withColumn("salary_after_tax",
                    col( "salary") - 2000)

df3.show()
```

Day2.py

```
# Databricks notebook source
# read csv file
file_path = "/Workspace/Users/vinaykryadav2002@gmail.com/Py_Spark/emp_data.csv"
df = spark.read.csv(file_path,header = True,
                    inferSchema=True)
df.show(1)


# COMMAND ----------

#check schema
df.printSchema()

# COMMAND ----------

# total rows count
df.count()

# COMMAND ----------

# summary
df.describe().show()

# COMMAND ----------

#select and filter
df.select("emp_name","salary").show(2)
df.filter(df.salary > 60000).show(2)

# COMMAND ----------

# Pyspark SQL Queries
df.createOrReplaceTempView('employees')

spark.sql("""
    SELECT city, ROUND(AVG(salary),2) AS avg_salary from employees group by city
    """).show(2)


# COMMAND ----------

# Handling Missing Values(Nulls)
from pyspark.sql.functions import col,isnull,isnan

# check null
df.filter(col("salary").isNull()).show()

#drop missing rows
df.dropna().show()

#fill missing salary with 0
df.fillna({"salary":0}).show()


# COMMAND ----------

#Task1 : CSV read karo (any sample data)
df = spark.read.csv(file_path,header = True,inferSchema=True)

# COMMAND ----------

#Task2 : SQL query likho: Find highest salary
df.createOrReplaceTempView("employee")
```

```
spark.sql("""
          select max(salary) as max_salary from employee
          """).show()


# COMMAND ----------

#Task3: Missing salary ko 0 se fill karo
from pyspark.sql.functions import col,isnull
df.fillna({"salary":0}).show()

# COMMAND ----------

#Task4: City wise employee count nikalo
df.createOrReplaceTempView("emp")
spark.sql("""
          select city, count(*) as emp_count from emp group by city
          """).show()
```

Day3.py

```
# Databricks notebook source
data = [
    (101, "Delhi", 5000),
    (102, "Mumbai", 7000),
    (103, "Delhi", 3000),
    (104, "Pune", 4000),
    (105, "Mumbai", 2000)
]
columns = ["id","city","amount"]
df = spark.createDataFrame(data,columns)
df.show()

# COMMAND ----------

#Group by
df.groupBy("city").sum("amount").show()

# COMMAND ----------

# multiple agg
from pyspark.sql.functions import avg,max,min
df.groupBy("city").agg(
    avg("amount").alias("avg_amount"),
    max("amount").alias("max_amount"),
    min("amount").alias("min_amount")
).show()

# COMMAND ----------

customer = [
    (1, "Vinay"),
    (2, "Amit"),
    (3, "Neha")
]
customer = spark.createDataFrame(customer,["id","name"])
customer.show()

orders = [
    (1, 5000),
    (2, 7000),
    (1, 3000),
    (3, 2000)
]
orders = spark.createDataFrame(orders,["id","amount"])
orders.show()
```

```
# COMMAND ----------

# Inner join
inner_join = customer.join(orders,on = "id",how = "inner")
inner_join.show()

# COMMAND ----------

left_join = customer.join(orders, on="id",how = "left")
left_join.show()

# COMMAND ----------

#right join
right_join = customer.join(orders,on = "id",how = "right")
right_join.show()

# COMMAND ----------

# customer total purchase
result = inner_join.groupBy("name").sum("amount")
result.show()

# COMMAND ----------

#sql
customer.createOrReplaceTempView("cust")
orders.createOrReplaceTempView("ord")
spark.sql("""
          select name, sum(amount) from cust c join ord o on c.id = o.id group by name
          """).show()

# COMMAND ----------

#Task 1: City wise total sales
df.groupBy("city").sum("amount").show()

# COMMAND ----------

# Task 2: Find customer who spent maximum
df.createOrReplaceTempView("tableA")
spark.sql("""
          select max(amount) from tableA group by id order by max(amount) desc limit 1
          """).show()

# COMMAND ----------

#Task3: Left join karo aur missing customers show karo
left_join = orders.join(customer,on = "id", how = "left")
left_join.filter(left_join.name.isNull()).show()

# COMMAND ----------
```

Day4.py

```
# Databricks notebook source
data = [
    ("Vinay", "Delhi", 5000),
    ("Vinay", "Delhi", 7000),
    ("Amit", "Mumbai", 6000),
    ("Amit", "Mumbai", 8000),
    ("Neha", "Pune", 4000)
]

columns = ["name", "city", "salary"]
df = spark.createDataFrame(data, columns)
```

```
df.show()


# COMMAND ----------

from pyspark.sql.window import Window
from pyspark.sql.functions import row_number,rank,dense_rank,lag,lead,sum,col


# COMMAND ----------

windowspace = Window.partitionBy("city").orderBy(col("salary").desc())

#Row Number
row_no = df.withColumn("row_num",row_number().over(windowspace))
#Rank
ranks = df.withColumn("rank",rank().over(windowspace))
#Dense Rank
denseRank = df.withColumn("denseRank",dense_rank().over(windowspace))
#Lag
Lag = df.withColumn("prev_salary",lag("salary",1).over(windowspace))

denseRank.show()



# COMMAND ----------

# Top Salary in Each City
df.withColumn("rnk",rank().over(windowspace)).filter("rnk = 1").show()

# COMMAND ----------

#Running Total (Cumulative Sum)
run_wind = Window.partitionBy("name").orderBy("salary")
df.withColumn("running",sum("salary").over(run_wind)).show()

# COMMAND ----------

df.withColumn("rank",dense_rank().over(windowspace)).filter("rank = 2").show()

# COMMAND ----------

#Task1: Each employee ka rank nikalo salary wise
t_wind1 = Window.partitionBy().orderBy(col("salary").desc())
emp_rank = df.withColumn("emp_rank",rank().over(t_wind1))
emp_rank.show()

# COMMAND ----------

#Task2: Find highest salary person in each city
t_wind2 = Window.partitionBy("city").orderBy(col("salary").desc())
h_salary = df.withColumn("high_salary_rank",rank().over(t_wind2)).filter("high_salary_rank = 1")

# COMMAND ----------

# Task3: Lag use karke previous salary show karo
t_wind3 = Window.partitionBy("name").orderBy("salary")
df.withColumn("prev_salary",lag("salary",1).over(t_wind3)).show()

# COMMAND ----------

t_wind4 = Window.partitionBy("city").orderBy(col("salary").desc())
df.withColumn("rank",dense_rank().over(t_wind4)).filter("rank = 2").show()
```

Day5.py

```
# Databricks notebook source
data = [
    (1, "2025-01-15", "Delhi", 5000),
    (2, "2025-02-10", "Mumbai", 7000),
    (3, "2025-02-20", "Delhi", 3000),
    (4, "2025-03-05", "Pune", 4000),
    (5, "2025-03-18", "Mumbai", 6000)
]

columns = ["order_id", "order_date", "city", "amount"]

df = spark.createDataFrame(data, columns)
df.show()


# COMMAND ----------

# Convert String to Date Type
from pyspark.sql.functions import to_date,col
df = df.withColumn("order_date",to_date(col("order_date"),"yyyy-MM-dd"))
df.printSchema()
df.show()


# COMMAND ----------

# Extract Year & Month
from pyspark.sql.functions import year,month,day
df = df.withColumn("Year",year("order_date"))\
    .withColumn("Month",month("order_date"))\
    .withColumn("Day",day("order_date"))
df.show()

# COMMAND ----------

#Monthly Sales Trend
df.groupBy("Year","Month").sum("amount").orderBy("Year","Month").show()

# COMMAND ----------

#Filter Data Between Dates
df.filter(col("order_date") >= "2025-02-01").show()

# COMMAND ----------

#Last 7 Days Data (Real Use Case)
from pyspark.sql.functions import current_date, date_sub,date_add,date_diff,date_format

df.filter(col("order_date") >= date_sub(current_date(),7)).show()

# COMMAND ----------

# Format date as Month-Year
df.withColumn("month-year",date_format("order_date","MMM-yyyy")).show()

# COMMAND ----------

monthly = df.groupBy("Month").sum("amount").show()

# COMMAND ----------

df.groupBy("city","Month").sum("amount").orderBy("city","Month").show()

# COMMAND ----------

df.filter(month("order_date")==3).show()

# COMMAND ----------
```

```
monthly = df.groupBy("Month").sum("amount")
monthly.orderBy(col("sum(amount)").desc()).show()

# COMMAND ----------
```

Day6.py

```
# Databricks notebook source
data = [
    (1,"Vinay","Delhi",5000),
    (2,"Amit","Mumbai",None),
    (3,"Neha","Pune",4000),
    (3,"Neha","Pune",4000),
    (4,None,"Delhi",3000)
]
columns = ["id","name","city","salary"]
df = spark.createDataFrame(data,columns)
df.show()

# COMMAND ----------

# check missing values
from pyspark.sql.functions import col
df.filter(col("salary").isNull()).show()

# COMMAND ----------

# fill missing value
df.fillna({"salary":0,"name":"unknown"}).show()

# COMMAND ----------

#Drop Rows with Null
df.dropna(subset=["name"]).show()

# COMMAND ----------

#Remove Duplicates
df.dropDuplicates(["id"]).show()

# COMMAND ----------

from pyspark.sql.functions import upper, trim
#Standardize Text
df.withColumn("city",upper("city")).show()
#Trim Spaces
df.withColumn("name",trim("name")).show()

# COMMAND ----------

#Create Clean Pipeline
clean_df = df.dropDuplicates().fillna({"salary":0,"name":"unknown"}).withColumn("city",upper("city"))
clean_df.show()


# COMMAND ----------

# DBTITLE 1,Write clean_df to supported path
#Save Clean Data
clean_df.write.csv("clean_df.csv",header = True)

# COMMAND ----------

df.groupBy("id").count().filter("count > 1").show()
```

Customer Sales Analysis.py

```
# Databricks notebook source
#Dataset Create
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Sale_project").getOrCreate()

data = [
    (1, "2025-01-10", "Vinay", "Delhi", 5000),
    (2, "2025-01-15", "Amit", "Mumbai", 7000),
    (3, "2025-02-10", "Vinay", "Delhi", 3000),
    (4, "2025-02-12", "Neha", "Pune", 4000),
    (5, "2025-03-05", "Amit", "Mumbai", 6000),
    (6, "2025-03-18", "Vinay", "Delhi", 8000)
]
columns = ["order_id","order_date","customer","city","amount"]

df = spark.createDataFrame(data,columns)
df.show()


# COMMAND ----------

#Data Cleaning
from pyspark.sql.functions import to_date,col,month,year,desc,asc

df = df.withColumn("order_date",to_date(col("order_date"),"yyyy-MM-dd"))

# COMMAND ----------

#Total Sales
df.groupBy().sum("amount").show()

# COMMAND ----------

#Top Customers
df.groupBy("customer").sum("amount").orderBy(col("sum(amount)").desc()).show()

# COMMAND ----------

#City Wise Sales
df.groupBy("city").sum("amount").orderBy(col("sum(amount)").desc()).show()

# COMMAND ----------

#Monthly Sales Trend
df = df.withColumn("Month",month("order_date")).withColumn("Year",year("order_date"))

df.groupBy("Year","Month").sum("amount").orderBy("Month").show()

# COMMAND ----------

#Repeat Customers
df.groupBy("customer").count().filter("count > 1").show()

# COMMAND ----------

#Top Customer per City
from pyspark.sql.window import Window
from pyspark.sql.functions import dense_rank,rank


windowSp = Window.partitionBy("City").orderBy(col("amount").desc())

df.withColumn("Rank",dense_rank().over(windowSp)).filter("Rank = 1").show()
# COMMAND ----------
```

PySpark Mock Interview - Corrected Solutions

```
Q1. Total Sales
df.groupBy().sum('amount').show()

Q2. City-wise Total Sales
df.groupBy('city').sum('amount').show()

Q3. Top 3 Customers by Spending
from pyspark.sql.functions import col
df.groupBy('customer').sum('amount')   .orderBy(col('sum(amount)').desc()).show(3)

Q4. Total Orders per Customer
df.groupBy('customer').count().show()

Q5. Monthly Sales Trend
from pyspark.sql.functions import year, month, col
df = df.withColumn('year', year(col('order_date')))        .withColumn('month', month(col('order
df.groupBy('year','month').sum('amount')   .orderBy('year','month').show()

Q6. Repeat Customers
df.groupBy('customer').count()   .filter(col('count') > 1).show()

Q7. Highest Spending City
df.groupBy('city').sum('amount')   .orderBy(col('sum(amount)').desc()).show(1)

Q8. Join & Segment-wise Sales
inner_join = df.join(customers, on='customer', how='inner')
inner_join.groupBy('segment').sum('amount').show()

Q9. Customers with No Orders
left_join = customers.join(df, on='customer', how='left')
left_join.filter(col('order_id').isNull()).show()

Q10. Each City Top Customer
from pyspark.sql.window import Window
from pyspark.sql.functions import dense_rank, sum

window_sp = Window.partitionBy('city').orderBy(col('amount').desc())
df.withColumn('rank', dense_rank().over(window_sp))   .filter(col('rank') == 1).show()

Q11. Running Total per Customer
window_sp = Window.partitionBy('customer').orderBy('order_date')
df.withColumn('running_total',
              sum('amount').over(window_sp)).show()

Q12. Data Cleaning Pipeline
from pyspark.sql.functions import upper
df.dropDuplicates()   .fillna({'name': 'unknown'})   .withColumn('city', upper(col('city')))
```