PySpark Interview Handbook — Batch 4

Problems 076–100
Generated: 2025-09-13 04:05:06Z (UTC)

Problem 076: 076 - Strings & Regex: Concat_ws challenge

Problem

Strings & Regex

Solution (PySpark)

```
from pyspark.sql import functions as F
res = users
if "email" in res.columns:
    res = res.withColumn("domain", F.regexp_extract("email", "@(.*)$", 1))
```

Problem 077: 077 - Complex Types: Structs challenge

Problem

Complex Types

Solution (PySpark)

```
from pyspark.sql import functions as F
res = orders
if "tags" in res.columns:
    res = res.withColumn("tag", F.explode_outer("tags"))
```

Problem 078: 078 - Performance & Tuning: Cache challenge

Problem

Performance & Tuning

Solution (PySpark)

```
from pyspark.sql import functions as F
res = sessions.repartition(200, "user_id").groupBy("user_id").agg(F.count("*").alias("cnt"))
```

Problem 079: 079 - DataFrame Basics: Drop challenge

Problem

DataFrame Basics

Solution (PySpark)

```
from pyspark.sql import functions as F
from pyspark.sql.window import Window
res = events.select("id", "user_id", "event_type", "value") \    .withColumn("value_norm", (F.col("value") - F.mean("va
assert "value_norm" in res.columns
```

Problem 080: 080 - Dates & Timestamps: To_date challenge

Problem

Dates & Timestamps

Solution (PySpark)

```
from pyspark.sql import functions as F
res = events.withColumn("day", F.to_date("ts")).groupBy("day").agg(F.count("*").alias("events"), F.avg("value").alias("
```

Problem 081: 081 - File IO & Formats: Json challenge

Problem

File IO & Formats

Solution (PySpark)

```
# Example write (commented):
# sessions.write.mode("overwrite").partitionBy("country").parquet("/path/out")
res = sessions
```

Problem 082: 082 - Performance & Tuning: Cache challenge

Problem

Performance & Tuning

Solution (PySpark)

```
from pyspark.sql import functions as F
res = sessions.repartition(200, "user_id").groupBy("user_id").agg(F.count("*").alias("cnt"))
```

Problem 083: 083 - Window Functions: Rank challenge

Problem

Window Functions

Solution (PySpark)

```
from pyspark.sql import functions as F
from pyspark.sql.window import Window
w = Window.partitionBy("user_id").orderBy("ts")
res = clicks.withColumn("rn", F.row_number().over(w)) \    .withColumn("prev_value", F.lag("value", 1).over(w)) \    .w
```

Problem 084: 084 - Performance & Tuning: Skew challenge

Problem

Performance & Tuning

Solution (PySpark)

```
from pyspark.sql import functions as F
res = clicks.repartition(200, "user_id").groupBy("user_id").agg(F.count("*").alias("cnt"))
```

Problem 085: 085 - Performance & Tuning: Checkpoint challenge

Problem

Performance & Tuning

Solution (PySpark)

```
from pyspark.sql import functions as F
res = products.repartition(200, "user_id").groupBy("user_id").agg(F.count("*").alias("cnt"))
```

Problem 086: 086 - Streaming (Structured): Watermark challenge

Problem

Streaming (Structured)

Solution (PySpark)

```
# streaming example would use readStream; here batch placeholder
res = logs
```

Problem 087: 087 - Complex Types: Maps challenge

Problem

Complex Types

Solution (PySpark)

```
from pyspark.sql import functions as F
res = transactions
if "tags" in res.columns:
    res = res.withColumn("tag", F.explode_outer("tags"))
```

Problem 088: 088 - Misc Utilities: Mappartitions challenge

Problem

Misc Utilities

Solution (PySpark)

```
res = sessions.hint("broadcast")
```

Problem 089: 089 - Joins: Right challenge

Problem

Joins

Solution (PySpark)

```
from pyspark.sql import functions as F
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
users = spark.createDataFrame([("u1","US"),("u2","IN")], ["user_id","country"])
res = transactions.join(F.broadcast(users), "user_id", "left")
```

Problem 090: 090 - Performance & Tuning: Repartition challenge

Problem

Performance & Tuning

Solution (PySpark)

```
from pyspark.sql import functions as F
res = logs.repartition(200, "user_id").groupBy("user_id").agg(F.count("*").alias("cnt"))
```

Problem 091: 091 - Misc Utilities: Broadcast joins challenge

Problem

Misc Utilities

Solution (PySpark)

```
res = products.hint("broadcast")
```

Problem 092: 092 - Streaming (Structured): Readstream challenge

Problem

Streaming (Structured)

Solution (PySpark)

```
# streaming example would use readStream; here batch placeholder
res = transactions
```

Problem 093: 093 - DataFrame Basics: Select challenge

Problem

DataFrame Basics

Solution (PySpark)

```
from pyspark.sql import functions as F
from pyspark.sql.window import Window
res = orders.select("id", "user_id", "event_type", "value") \    .withColumn("value_norm", (F.col("value") - F.mean("va
assert "value_norm" in res.columns
```

Problem 094: 094 - UDFs & Pandas UDFs: Udf challenge

Problem

UDFs & Pandas UDFs

Solution (PySpark)

```
from pyspark.sql import functions as F
from pyspark.sql.types import DoubleType
@F.udf(DoubleType())
def score(x): return float(x)*1.1 if x is not None else None
res = clicks.withColumn("score", score("value"))
```

Problem 095: 095 - Misc Utilities: Broadcast joins challenge

Problem

Misc Utilities

Solution (PySpark)

```
res = products.hint("broadcast")
```

Problem 096: 096 - Dates & Timestamps: Date_trunc challenge

Problem

Dates & Timestamps

Solution (PySpark)

```
from pyspark.sql import functions as F
res = clicks.withColumn("day", F.to_date("ts")).groupBy("day").agg(F.count("*").alias("events"), F.avg("value").alias("
```

Problem 097: 097 - Graph-ish / Hierarchical: Recursive-like with joins challenge

Problem

Graph-ish / Hierarchical

Solution (PySpark)

```
res = users.hint("broadcast")
```

Problem 098: 098 - Joins: Left challenge

Problem

Joins

Solution (PySpark)

```
from pyspark.sql import functions as F
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
users = spark.createDataFrame([("u1","US"),("u2","IN")], ["user_id","country"])
res = clicks.join(F.broadcast(users), "user_id", "left")
```

Problem 099: 099 - Performance & Tuning: Aqe challenge

Problem

Performance & Tuning

Solution (PySpark)

```
from pyspark.sql import functions as F
res = users.repartition(200, "user_id").groupBy("user_id").agg(F.count("*").alias("cnt"))
```

Problem 100: 100 - Streaming (Structured): Readstream challenge

Problem

Streaming (Structured)

Solution (PySpark)

```
# streaming example would use readStream; here batch placeholder
res = clicks
```