APACHE SPARK EXERCISES

DAG • Broadcast Joins • Transformations • Actions • Partitions

---

DOMAIN

Ride-Hailing Platform Analytics (Uber / Ola-like)

DATASET 1 — RIDES (Large Fact Table)

```
rides_data = [
    ("R001","U001","Hyderabad",12.5,240,"Completed"),
    ("R002","U002","Delhi",8.2,180,"Completed"),
    ("R003","U003","Mumbai",15.0,300,"Cancelled"),
    ("R004","U004","Bangalore",5.5,120,"Completed"),
    ("R005","U005","Hyderabad",20.0,360,"Completed"),
    ("R006","U006","Delhi",25.0,420,"Completed"),
    ("R007","U007","Mumbai",7.5,150,"Completed"),
    ("R008","U008","Bangalore",18.0,330,"Completed"),
    ("R009","U009","Delhi",6.0,140,"Cancelled"),
    ("R010","U010","Hyderabad",10.0,200,"Completed")
]

rides_cols = [
    "ride_id",
    "user_id",
    "city",
    "distance_km",
    "duration_seconds",
    "status"
]
```

```
rides_df = spark.createDataFrame(rides_data, rides_cols)
```

DATASET 2 — CITY SURGE MULTIPLIERS (Small Lookup)

```
surge_data = [
    ("Hyderabad",1.2),
    ("Delhi",1.5),
    ("Mumbai",1.8),
    ("Bangalore",1.3)
]

surge_cols = ["city","surge_multiplier"]

surge_df = spark.createDataFrame(surge_data, surge_cols)
```

EXERCISE SET 1 — TRANSFORMATIONS vs ACTIONS

Exercise 1.1

Create a transformation pipeline that:

- Filters only `Completed` rides
- Selects `ride_id`, `city`, `distance_km`

Tasks:

- Do not trigger any action
- Explain whether Spark executed anything

Exercise 1.2

Trigger a single action on the pipeline.

Tasks:

- Identify which line caused execution
- Explain why previous lines did not execute

EXERCISE SET 2 — DAG & LINEAGE

Exercise 2.1

Create a transformation chain with:

- Multiple filters
- A column selection

Tasks:

- Run `explain(True)`
- Identify:
    - Logical plan
    - Optimized logical plan
    - Physical plan

Exercise 2.2

Reorder transformations (filter after join vs before join).

Tasks:

- Compare DAGs
- Identify which plan is more efficient and why

EXERCISE SET 3 — PARTITIONS & SHUFFLE

Exercise 3.1

Check the number of partitions of `rides_df`.

Tasks:

- Repartition into 4 partitions

- Coalesce into 1 partition
- Observe number of output files when writing to Parquet

Exercise 3.2

Repartition rides by `city`.

Tasks:

- Run `explain(True)`
- Identify whether a shuffle is introduced

EXERCISE SET 4 — JOIN WITHOUT BROADCAST (BAD DAG)

Exercise 4.1

Join `rides_df` with `surge_df` on `city` **without using broadcast**.

Tasks:

- Run `explain(True)`
- Identify:
    - Join type
    - Exchange operators
    - Sort operations
    - Stage boundaries

Exercise 4.2

Apply a filter (`distance_km > 10`) before the join.

Tasks:

- Observe whether shuffle is removed
- Explain why or why not

EXERCISE SET 5 — BROADCAST JOIN (GOOD DAG)

Exercise 5.1

Apply a broadcast hint to `surge_df`.

Tasks:

- Run `explain(True)`
- Identify:
  - Join type
  - BroadcastExchange
  - Disappearance of shuffles

Exercise 5.2

Compare physical plans from:

- Exercise 4.1
- Exercise 5.1

Tasks:

- List operators that disappeared
- Explain performance impact

EXERCISE SET 6 — DAG INTERPRETATION

Exercise 6.1

From the physical plan:

- Identify all expensive operators
- Classify them as CPU, memory, or network heavy

Exercise 6.2

Explain why Spark defaults to `SortMergeJoin`.

EXERCISE SET 7 — ACTION-DRIVEN EXECUTION

Exercise 7.1

Create a long transformation pipeline without any action.

Tasks:

- Explain what Spark has done so far

Exercise 7.2

Trigger different actions (`count` , `show` , `write`) separately.

Tasks:

- Observe whether Spark recomputes the DAG
- Explain behavior

EXERCISE SET 8 — THINKING QUESTIONS (WRITTEN)

1. Why does broadcast remove shuffle from the DAG?
2. Why does repartition always introduce shuffle?
3. Why is coalesce cheaper than repartition?
4. Why does Spark delay execution until an action?
