🔷 PYSPARK EXERCISES – LARGE DATASET

Dataset: `orders_large_bad.csv` / `orders_large_bad.json`

📌 CONTEXT

You are a **Data Engineer** working with **raw production data** ingested from multiple upstream systems.

The dataset:

- Is large (hundreds of thousands of rows)
- Has bad formatting
- Has data quality issues
- Is NOT analytics-ready

Your job is to **ingest, clean, optimize, analyze, and store** this data using PySpark.

📂 INPUT DATA

Files provided:

- `orders_large_bad.csv`
- `orders_large_bad.json`

Columns:

```
order_id
customer_id
city
category
product
amount
order_date
status
```

🔹 PHASE 1 — INGESTION & FIRST INSPECTION

Exercises

1. Read the CSV file into a DataFrame
2. Disable schema inference and read everything as string
3. Print schema and record count
4. Display 20 random rows
5. Identify **at least 5 data quality issues** by observation
6. Read the JSON file and compare schema and row count with CSV

🔹 PHASE 2 — SCHEMA ENFORCEMENT & VALIDATION

Exercises

7. Define an explicit schema using `StructType`
8. Re-read the CSV using the defined schema
9. Identify rows that fail schema expectations
10. Explain why schema inference is dangerous at scale

🔹 PHASE 3 — STRING CLEANING & STANDARDIZATION

Exercises

11. Trim leading and trailing spaces from all string columns
12. Standardize `city`, `category`, and `product` values

13. Convert all categorical columns to a consistent case
14. Identify how many distinct city values existed before vs after cleaning

---

🔷 PHASE 4 — AMOUNT CLEANING (CRITICAL)

Exercises

15. Identify invalid values in the `amount` column
16. Remove commas from numeric strings
17. Convert `amount` to IntegerType safely
18. Handle empty, null, and invalid values explicitly
19. Count how many records were affected during amount cleaning

---

🔷 PHASE 5 — DATE PARSING & NORMALIZATION

Exercises

20. Identify all date formats present in `order_date`
21. Parse valid dates into `DateType`
22. Handle invalid dates gracefully
23. Create a clean `order_date_clean` column
24. Count records with invalid dates

🔷 PHASE 6 — BUSINESS FILTERING & DEDUPLICATION

Exercises

25. Identify duplicate `order_id` values
26. Remove duplicate orders safely

27. Keep only records with status = `Completed`
28. Validate record counts before and after filtering

🔹 PHASE 7 — PERFORMANCE & PARTITION AWARENESS

Exercises

29. Check the default number of partitions
30. Run a heavy `groupBy` and observe execution time
31. Use `explain(True)` to identify shuffle stages
32. Repartition the DataFrame by `city`
33. Compare execution plans before and after repartition

🔹 PHASE 8 — ANALYTICS ON LARGE DATA

Exercises

34. Calculate total revenue per city
35. Calculate total revenue per category
36. Calculate total revenue per product
37. Identify top 10 products by revenue
38. Calculate average order value per city

🔹 PHASE 9 — WINDOW FUNCTIONS (BIG DATA SAFE)

Exercises

39. Rank cities by total revenue
40. Rank products within each category by revenue

41. Identify the top product per category
42. Identify top 3 cities using window functions

🔷 PHASE 10 — CACHING & REUSE

Exercises

43. Identify DataFrames reused multiple times
44. Apply caching strategically
45. Re-run analytics and observe performance
46. Unpersist when cache is no longer needed
47. Explain why over-caching is dangerous

🔷 PHASE 11 — FILE FORMAT STRATEGY

Exercises

48. Write the cleaned order-level dataset to **Parquet**
49. Partition the Parquet output by `city`
50. Write aggregated analytics to **ORC**
51. Read both formats back and validate schema
52. Compare number of output files generated

🔷 PHASE 12 — DEBUGGING & FAILURE SCENARIOS

Exercises

53. Explain why the following line breaks pipelines:

```
df = df.filter(df.amount > 50000).show()
```

54. Create a scenario that produces a `NoneType` error
55. Identify a transformation that causes a wide shuffle
56. Explain how you would debug a slow Spark job

🔷 PHASE 13 — FINAL VALIDATION

Exercises

57. Validate no nulls in critical columns
58. Confirm correct data types for all columns
59. Validate final record count
60. Document three optimization decisions you made
