🧠 BUSINESS CONTEXT

You are a **Data Engineer** working for a **logistics & delivery company** operating across multiple cities.

The company wants to:

- Track delivery performance
- Optimize operational analytics
- Prepare clean datasets for dashboards
- Improve Spark job performance
- Design scalable storage formats

Your task is to **build an optimized PySpark data pipeline** from raw operational data.

📂 DATASETS PROVIDED (RAW, INCONSISTENT, REALISTIC)

DATASET 1 — DELIVERY TRANSACTIONS (CSV)

```
delivery_data = [
    ("DLV001","Delhi ","D001","Delivered","120","2024-01-05 10:30"),
    ("DLV002","Mumbai","D002","Delivered","90","05/01/2024 11:00"),
    ("DLV003","Bangalore","D003","In Transit","200","2024/01/06 09:45"),
    ("DLV004","Delhi","D004","Cancelled","","2024-01-07 14:00"),
    ("DLV005","Chennai","D002","Delivered","invalid","2024-01-08 16:20"),
    ("DLV006","Mumbai","D005","Delivered",None,"2024-01-08 18:10"),
    ("DLV007","Delhi","D001","Delivered","140","09-01-2024 12:30"),
    ("DLV008","Bangalore","D003","Delivered","160","2024-01-09 15:45"),
    ("DLV009","Mumbai","D004","Delivered","110","2024-01-10 13:20"),
    ("DLV009","Mumbai","D004","Delivered","110","2024-01-10 13:20")
]
```

Columns

```
delivery_id
city
driver_id
status
delivery_time_minutes
delivery_timestamp
```

DATASET 2 — DRIVER MASTER (JSON)

```
driver_data = [
    ("D001","Ravi","Senior"),
    ("D002","Amit","Junior"),
    ("D003","Sneha","Senior"),
    ("D004","Karan","Junior"),
    ("D005","Neha","Senior")
]
```

DATASET 3 — CITY ZONE LOOKUP (REFERENCE)

```
city_zone_data = [
    ("Delhi","North"),
    ("Mumbai","West"),
    ("Bangalore","South"),
    ("Chennai","South")
]
```

🎯 PROJECT OBJECTIVES

PHASE 1 — SCHEMA DESIGN & INGESTION

Topics:

StructType, StructField, data types, corrupt handling

Tasks

1. Define explicit schemas for all datasets
2. Load raw delivery data using schema enforcement
3. Identify and flag corrupt records
4. Validate schema correctness

PHASE 2 — DATA CLEANING & STANDARDIZATION

Topics:
Column ops, filter, withColumn, replace, date handling

Tasks

5. Trim all string columns
6. Standardize `status` values
7. Convert `delivery_time_minutes` to IntegerType
8. Handle invalid and null delivery times
9. Parse multiple timestamp formats into TimestampType
10. Remove duplicate delivery IDs

PHASE 3 — BUSINESS FILTERING

Topics:
Filters, transformations

Tasks

11. Keep only `Delivered` deliveries
12. Remove cancelled and in-transit deliveries
13. Validate record counts before and after filtering

PHASE 4 — DATA ENRICHMENT & JOINS

Topics:
Joins, broadcast, explain()

Tasks

14. Join delivery data with driver master
15. Join enriched data with city zone lookup
16. Use broadcast join where appropriate
17. Explain join strategy using `explain(True)`

PHASE 5 — ANALYTICS & WINDOW FUNCTIONS

Topics:
Aggregations, Window, ranking

Tasks

18. Average delivery time per city
19. Average delivery time per driver
20. Rank drivers by performance within each city
21. Identify fastest driver per zone
22. Identify top 2 drivers per city

PHASE 6 — PERFORMANCE OPTIMIZATION

Topics:
Cache, persist, repartition, shuffles

Tasks

23. Identify DataFrames reused multiple times
24. Apply caching appropriately
25. Compare execution plans with and without cache
26. Repartition data by city

27. Explain why repartitioning improves performance

PHASE 7 — FILE FORMAT STRATEGY

Topics:
Parquet, ORC, Avro (conceptual)

Tasks

28. Write cleaned delivery data to Parquet
29. Write aggregated analytics to ORC
30. Compare file output structure
31. Explain why Avro is suitable for future real-time tracking

---

PHASE 8 — DEBUGGING & ERROR ANALYSIS

Topics:
Debugging, common Spark errors

Tasks

32. Identify potential NoneType errors
33. Identify schema mismatch risks
34. Debug an intentionally broken transformation
35. Use explain plan to find inefficient operations
