Scenario: Large-Scale Order Processing System

You are a **Data Engineer** at a retail analytics company.
Your system receives **raw daily order data** from multiple sources in CSV format.

The data is known to be:

- Large (hundreds of thousands to millions of records per day)
- Messy (inconsistent formats, nulls, duplicates)
- Required for both **analytics** and **reporting**

You are provided with a file:

```
orders_raw.csv
```

Schema (Raw)

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

🧪 Assessment Tasks

Section 1 — Data Ingestion & Schema

1. Read the CSV file using PySpark ensuring the job does not fail due to bad data.
2. Explain why reading all columns as `StringType` is preferred initially.
3. Print schema and total record count.

Section 2 — Data Cleaning & Validation

4. Clean leading/trailing spaces from string columns.
5. Standardize `city`, `category`, and `product` values.
6. Convert `amount` to integer safely, handling invalid values.
7. Parse `order_date` supporting multiple date formats.
8. Identify and handle invalid or null records.

Section 3 — Business Rules

9. Remove duplicate records based on `order_id`.
10. Filter only records with `status = Completed`.
11. Validate record counts before and after filtering.

Section 4 — Performance & Optimization

12. Identify operations that cause shuffles.
13. Use `explain(True)` to analyze the execution plan.
14. Apply repartitioning to optimize aggregations.
15. Justify where caching should be applied and why.

Section 5 — Analytics

16. Calculate total revenue per city.
17. Calculate total revenue per category.
18. Identify top 5 products by revenue.
19. Calculate average order value per city.

Section 6 — Window Functions

20. Rank cities by total revenue.
21. Rank products within each category by revenue.
22. Identify the top product per category.

Section 7 — Storage Strategy

23. Write the cleaned dataset to Parquet partitioned by `city`.
24. Write aggregated analytics to ORC.

25. Explain why CSV is not suitable for analytics output.

Section 8 — Debugging & Reasoning

26. Explain why the following line causes failure:

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

27. Describe how you would debug a slow Spark job.
28. Identify risks of over-caching DataFrames.
