REAL-TIME CASE STUDY #3

E-Commerce Marketplace Orders, Customers & Seller Performance Analytics

BUSINESS CONTEXT

You are a **Data Engineer** working for a **large e-commerce marketplace** (Amazon / Flipkart / Meesho style).

Business wants to:

- Identify **high-value customers**
- Measure **seller performance**
- Understand **category-wise revenue**
- Detect **inactive sellers and customers**
- Optimize **order fulfillment**

Data is coming from **multiple internal systems**, each with **data quality issues**.

Your task is to **clean, integrate, analyze, and optimize** the data using **PySpark**.

DATA SOURCES PROVIDED

You are given **5 raw datasets**.

DATASET 1 — CUSTOMER MASTER (CORRUPTED)

```
raw_customers = [
    ("C001","Rahul","29","Bangalore","Electronics,Fashion"),
    ("C002","Sneha","Thirty Two","Delhi","Fashion"),
    ("C003","Aman",None,"Mumbai",["Home","Electronics"]),
    ("C004","Pallavi","27","Pune","Electronics|Beauty"),
    ("C005","", "35","Chennai",None)
]
```

Known Issues

- Age mixed formats
- Interests as string / array / multiple delimiters
- Missing names
- Null interests

DATASET 2 — SELLER MASTER

```
raw_sellers = [
    ("S001","TechWorld","Electronics","2019-06-01"),
    ("S002","FashionHub","Fashion","01/07/2020"),
    ("S003","HomeEssentials","Home","2018/09/15"),
    ("S004","BeautyStore","Beauty","invalid_date")
]
```

Known Issues

- Seller onboarding date in multiple formats
- Invalid dates

DATASET 3 — PRODUCT CATALOG

```
raw_products = [
    ("P001","Laptop","Electronics","S001","55000"),
    ("P002","Headphones","Electronics","S001","2500"),
    ("P003","T-Shirt","Fashion","S002","1200"),
    ("P004","Sofa","Home","S003","45000"),
    ("P005","Face Cream","Beauty","S004","800")
]
```

Known Issues

- Price as string
- Category repetition
- Foreign key to seller

DATASET 4 — ORDERS DATA

```
raw_orders = [
    ("O001","C001","P001","2024-01-05","Delivered","55000"),
    ("O002","C002","P003","05/01/2024","Cancelled","0"),
    ("O003","C003","P004","2024/01/06","Delivered","45000"),
    ("O004","C004","P005","invalid_date","Delivered","800"),
    ("O005","C001","P002","2024-01-10","Delivered","2500"),
    ("O006","C005","P003","2024-01-12","Delivered","1200")
]
```

Known Issues

- Multiple date formats
- Invalid dates
- Cancelled orders
- Revenue as string

DATASET 5 — CUSTOMER ACTIVITY LOGS

```
raw_activity = [
    ("C001","search,view,add_to_cart","{'device':'mobile'}",180),
    ("C002",["search","view"],"device=laptop",90),
    ("C003","search|view|purchase",None,120),
    ("C004",None,"{'device':'tablet'}",60),
    ("C005","search","{'device':'mobile'}",30)
]
```

Known Issues

- Actions in multiple formats
- Metadata as JSON-like strings
- Missing actions

BUSINESS QUESTIONS TO ANSWER

You must complete **all sections**.

PART A — DATA CLEANING & STRUCTURING

1. Design **explicit schemas** for all datasets
2. Normalize:
    - Age
    - Prices
    - Dates
3. Convert interests and actions into arrays
4. Handle missing and invalid records gracefully
5. Produce clean DataFrames:
    - `customers_df`
    - `sellers_df`
    - `products_df`
    - `orders_df`
    - `activity_df`

PART B — DATA INTEGRATION (JOINS)

6. Join orders with products
7. Join products with sellers
8. Join orders with customers
9. Decide which table(s) should be **broadcast**
10. Prove your decision using `explain(True)`
11. Eliminate orphan records

PART C — ANALYTICS & AGGREGATIONS

12. Total revenue per category
13. Total revenue per seller
14. Total orders per customer
15. Average order value per customer
16. Identify sellers with **zero delivered orders**

PART D — WINDOW FUNCTIONS

17. Rank customers by total spend (overall)
18. Rank sellers by revenue **within each category**
19. Calculate running revenue per day
20. Identify top 2 products per category by revenue

PART E — UDF (ONLY IF REQUIRED)

21. Classify customers into spending tiers:
    - High
    - Medium
    - Low

Rules:

- Prefer built-in functions
- Use UDF only if unavoidable
- Justify your choice

PART F — SORTING & ORDERING

22. Sort categories by total revenue (descending)
23. Sort sellers by revenue within category
24. Explain why sorting caused a shuffle

PART G — SET OPERATIONS

Create two DataFrames:

- Customers who **placed orders**
- Customers who **were active (search/view)**

25. Find customers who were active but never ordered
26. Find customers who ordered and were active
27. Explain why set operations differ from joins

PART H — DAG & PERFORMANCE ANALYSIS

28. Run `explain(True)` for:
    - Product → Seller join
    - Window ranking
    - Sorting
29. Identify:
    - Shuffles
    - Broadcast joins
    - Sort stages
30. Suggest one performance improvement
