CASE STUDY 2

Title: Customer Purchase Behavior & Loyalty Analysis using PySpark

A retail company wants to understand how customers behave across cities and product categories.
They are not interested only in revenue, but in **customer patterns**:

- Who are the high-value customers?
- Which customers are loyal?
- Which cities have repeat buyers?
- Which categories drive long-term engagement?

They provide a single file:

```
orders.csv
```

This is the same file used in Case Study 1.
No new datasets are assumed.
No extra corruption.
Same raw production-style CSV.

Columns:

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

Your job is to build a PySpark pipeline that transforms this transactional data into **customer intelligence**.

This case study focuses more on:

- Window functions
- Aggregations
- Customer segmentation
- Analytical thinking
- Performance design

Not just cleaning.

PHASE 1 – Ingestion & Cleaning

Use the same cleaning logic as Case Study 1:

1. Read orders.csv as all StringType.
2. Trim text columns.
3. Normalize city, category, product.
4. Clean amount:
    - Remove commas
    - Convert to IntegerType
    - Handle invalid values safely.
5. Parse order_date into DateType → `order_date_clean` .
6. Remove duplicate order_id.
7. Keep only `Completed` orders.

From this point onward, the dataset is considered **clean_orders_df**.

PHASE 2 – Customer Metrics

Compute the following for each customer:

1. Total number of orders.
2. Total spending.
3. Average order value.
4. First purchase date.
5. Last purchase date.
6. Number of distinct cities ordered from.
7. Number of distinct categories ordered from.

These define the **customer profile**.

PHASE 3 – Customer Segmentation

Create customer segments using business logic:

```
Total Spend >= 200000 AND Orders >= 5  → "VIP"
Total Spend >= 100000                  → "Premium"
Else                                   → "Regular"
```

Add a column:

```
customer_segment
```

Count customers in each segment.

PHASE 4 – Window Functions

Using Window functions:

1. Rank customers by total spending (overall).
2. Rank customers inside each city by total spending.
3. Identify top 3 customers per city.
4. Identify top 10 customers across all cities.

This phase must use:

```
Window.partitionBy()
```

PHASE 5 – Customer Loyalty Analysis

Define loyalty:

A loyal customer is one who:

- Has purchases on at least 3 different dates
- Has ordered from at least 2 different categories

Tasks:

1. Identify loyal customers.
2. Count loyal customers per city.
3. Compare loyal vs non-loyal customer revenue contribution.

PHASE 6 – Time-Based Analysis

Using order_date_clean:

1. Compute monthly revenue per city.
2. Compute monthly order count per category.
3. Identify growth or decline trends.

This introduces:

- Date functions
- Time-series thinking

PHASE 7 – Performance Engineering

1. Identify which DataFrames are reused.
2. Apply caching.
3. Use explain(True) on:
    - Customer aggregation
    - Window ranking
4. Identify shuffle stages.
5. Justify any repartitioning strategy.

PHASE 8 – Broadcast Join (Light Use)

Create a small lookup:

```
segment_code,segment_label
1,VIP
2,Premium
3,Regular
```

Map:

```
VIP     → 1
Premium → 2
Regular → 3
```

Tasks:

1. Create this as a small DataFrame.
2. Join with customer segmentation output.
3. Force broadcast join.
4. Verify BroadcastHashJoin in plan.

PHASE 9 – Sorting & Set Operations

1. Sort customers by:

- Total spend descending
- Order count descending

2. Create two sets:

   - Customers who bought Electronics
   - Customers who bought Grocery

3. Find:

   - Customers in both sets
   - Customers in only one set

PHASE 10 – Storage Strategy

1. Write customer master dataset to:

```
Parquet
```

Partitioned by:

```
customer_segment
```

2. Write monthly analytics to:

```
ORC
```

3. Read back and validate.

PHASE 11 – Debugging

Explain why this is dangerous:

```
df = df.groupBy("customer_id").sum("amount").show()
```

Explain:

- What df becomes
- Why pipeline breaks
- Correct approach
