
New Dataset: Retail Transactions

Sample Data (Create this as `retail_data.csv`)

```
TransactionID,Customer,City,Product,Category,Quantity,UnitPrice,TotalPrice,TransactionDa

T1001,Ali,Mumbai,Laptop,Electronics,1,70000,70000,2024-01-15,Card
T1002,Neha,Bangalore,Tablet,Electronics,2,30000,60000,2024-01-20,UPI
T1003,Ravi,Hyderabad,Desk,Furniture,1,15000,15000,2024-02-10,Net Banking
T1004,Zoya,Delhi,Chair,Furniture,4,5000,20000,2024-02-12,Card
T1005,Karan,Mumbai,Phone,Electronics,1,50000,50000,2024-02-15,Card
T1006,Farah,Delhi,Mouse,Electronics,3,1000,3000,2024-02-18,Cash
```

Task Set – PySpark Hands-On (No DLT)

Basics

1. Load `retail_data.csv` into a PySpark DataFrame and display schema.
2. Infer schema as False — then manually cast columns.

Data Exploration & Filtering

3. Filter transactions where `TotalPrice > 40000`.
4. Get unique cities from the dataset.
5. Find all transactions from "Delhi" using `.filter()` and `.where()`.

Data Manipulation

6. Add a column `DiscountedPrice` = `TotalPrice` - 10%.
7. Rename `TransactionDate` to `TxnDate`.
8. Drop the column `UnitPrice`.

Aggregations

9. Get total sales by city.
10. Get average unit price by category.
11. Count of transactions grouped by PaymentMode.

Window Functions

12. Use a window partitioned by City to rank transactions by `TotalPrice`.
13. Use lag function to get previous transaction amount per city.

Joins

14. Create a second DataFrame `city_region`:

```
City,Region
Mumbai,West
Delhi,North
```

```
Bangalore,South
Hyderabad,South
```

15. Join with main DataFrame and group total sales by Region.

▯ Nulls and Data Cleaning

16. Introduce some nulls and replace them with default values.
17. Drop rows where `Quantity` is null.
18. Fill null `PaymentMode` with "Unknown".

▯ Custom Functions

19. Write a UDF to label orders:

```python
def label_order(amount):
    if amount > 50000: return "High"
    elif amount >= 30000: return "Medium"
    else: return "Low"
```

Apply this to classify `TotalPrice`.

▯ Date & Time

20. Extract year, month, and day from `TxnDate`.
21. Filter transactions that happened in February.

▯ Union & Duplicate Handling

22. Duplicate the DataFrame using `union()` and remove duplicates.
