DOMAIN

Food Delivery Platform Analytics (Swiggy / Zomato-like)

Each row represents **one food order**.

DATASET — FOOD ORDERS

Step 1: Create the dataset

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Food Delivery Analytics") \
    .getOrCreate()


orders_data = [
    ("O001","North","Delhi","Rest-01","Pizza","2024-02-01",450,35),
    ("O002","North","Delhi","Rest-01","Burger","2024-02-01",250,25),
    ("O003","North","Chandigarh","Rest-02","Pasta","2024-02-02",350,30),
    ("O004","South","Bangalore","Rest-03","Pizza","2024-02-01",500,40),
    ("O005","South","Chennai","Rest-04","Burger","2024-02-02",220,20),
    ("O006","South","Bangalore","Rest-03","Pasta","2024-02-03",380,32),
    ("O007","East","Kolkata","Rest-05","Pizza","2024-02-01",420,38),
    ("O008","East","Kolkata","Rest-05","Burger","2024-02-02",260,26),
    ("O009","East","Patna","Rest-06","Pasta","2024-02-03",300,28),
    ("O010","West","Mumbai","Rest-07","Pizza","2024-02-01",520,42),
    ("O011","West","Mumbai","Rest-07","Burger","2024-02-02",280,27),
    ("O012","West","Pune","Rest-08","Pasta","2024-02-03",340,31),
    ("O013","North","Delhi","Rest-01","Pizza","2024-02-04",480,37),
    ("O014","South","Chennai","Rest-04","Pizza","2024-02-04",510,41),
    ("O015","East","Patna","Rest-06","Burger","2024-02-04",240,24),
    ("O016","West","Pune","Rest-08","Pizza","2024-02-04",500,39),
    ("O017","North","Chandigarh","Rest-02","Burger","2024-02-05",260,26),
    ("O018","South","Bangalore","Rest-03","Burger","2024-02-05",290,29),
    ("O019","East","Kolkata","Rest-05","Pasta","2024-02-05",360,33),
    ("O020","West","Mumbai","Rest-07","Pasta","2024-02-05",390,34),
```

```
    ("O021","North","Delhi","Rest-01","Pasta","2024-02-06",370,30),
    ("O022","South","Chennai","Rest-04","Pasta","2024-02-06",330,29),
    ("O023","East","Patna","Rest-06","Pizza","2024-02-06",460,36),
    ("O024","West","Pune","Rest-08","Burger","2024-02-06",270,26)
]

columns = [
    "order_id","region","city","restaurant_id",
    "food_item","order_date","amount","delivery_time_min"
]

df_orders = spark.createDataFrame(orders_data, columns)
df_orders.show(5)
df_orders.printSchema()
```

EXERCISE SET 1 — SELECT OPERATIONS

1. Select only `order_id`, `region`, `food_item`, `amount`
2. Rename `amount` to `order_value`
3. Create a new column `amount_in_hundreds`
4. Select distinct combinations of `region` and `food_item`
5. Reorder columns in a logical reporting format
6. Create a column `order_day` extracted from `order_date`

EXERCISE SET 2 — FILTER OPERATIONS

1. Filter orders where amount > 400
2. Filter only `Pizza` orders
3. Filter orders from `Delhi` and `Mumbai`
4. Filter orders with delivery time greater than 35 minutes
5. Apply multiple conditions using `AND` and `OR`
6. Apply filters in different orders and compare `explain(True)`
7. Identify which filters are pushed down by Spark

EXERCISE SET 3 — TRANSFORMATIONS vs ACTIONS

1. Build a pipeline with:
    - select
    - filter
    - derived column
2. Do not call any action
3. Explain what Spark has done so far
4. Trigger `count()` and observe execution
5. Trigger `show()` and compare behavior

EXERCISE SET 4 — PARTITIONS & FILE LAYOUT

1. Check the number of partitions of `df_orders`
2. Repartition the DataFrame into 4 partitions
3. Coalesce the DataFrame into 1 partition
4. Write repartitioned data to Parquet and count files
5. Write coalesced data to Parquet and count files
6. Explain why file counts differ

EXERCISE SET 5 — GROUPBY & AGGREGATE FUNCTIONS

1. Total revenue per region
2. Average order amount per food item
3. Maximum order amount per city

4. Minimum delivery time per restaurant
5. Count number of orders per region
6. Total revenue per restaurant
7. Region + food item wise total revenue
8. City wise average delivery time
9. Identify regions with revenue above a threshold
10. Use `explain(True)` and identify shuffle operators

EXERCISE SET 6 — WINDOW FUNCTIONS (OVER)

(Students must import Window and functions)

1. Compute running total of revenue per region ordered by date
2. Rank orders by amount within each region
3. Assign row numbers per restaurant based on delivery time
4. Use dense rank to rank food items per region by revenue
5. Identify top 2 highest value orders per region
6. Compare `rank`, `dense_rank`, and `row_number` outputs
7. Calculate cumulative delivery time per restaurant

EXERCISE SET 7 — GROUPBY vs WINDOW (CONCEPTUAL)

1. Calculate total revenue per region using GroupBy
2. Calculate total revenue per region using Window
3. Compare:
    - Row count
    - Output structure
    - Use case
4. Explain why Window does not reduce rows

EXERCISE SET 8 — DAG & PERFORMANCE ANALYSIS

1. Run `explain(True)` for:
    - Simple select
    - Filter
    - GroupBy aggregation
    - Window function
2. Identify:
    - Exchange operators
    - Sort operations
    - Stage boundaries
3. Explain why window functions require sorting
4. Identify expensive operations in each DAG

EXERCISE SET 9 — THINKING QUESTIONS (WRITE ANSWERS)

1. Why does GroupBy introduce shuffle?
2. Why does Window not reduce rows?
3. Why does repartition always cause shuffle?
4. Why is coalesce cheaper than repartition?
5. Why does Spark delay execution until an action?
6. When would you avoid window functions?
