USE CASE

Online Food Delivery Orders Dataset

Think Swiggy / Zomato / Uber Eats analytics.

DATASET: Food Delivery Orders (40 Records)

Schema

- order_id
- customer_name
- city
- restaurant
- cuisine
- order_amount
- delivery_time_minutes
- payment_mode
- order_status

Create the Dataset in PySpark

```
data = [
    ("O001","Amit","Hyderabad","Spice Hub","Indian",450,35,"UPI","Deliver
    ("O002","Neha","Bangalore","Pizza Town","Italian",650,40,"Card","Deli
    ("O003","Rahul","Delhi","Burger Zone","American",520,30,"Cash","Deliv
    ("O004","Pooja","Mumbai","Sushi Bar","Japanese",1200,55,"UPI","Cancel
    ("O005","Arjun","Chennai","Curry Leaf","Indian",380,28,"UPI","Deliver
    ("O006","Sneha","Hyderabad","Pasta Street","Italian",700,45,"Card","D
    ("O007","Karan","Delhi","Taco Bell","Mexican",540,33,"UPI","Delivered
    ("O008","Riya","Bangalore","Dragon Bowl","Chinese",600,38,"Wallet","D
    ("O009","Vikas","Mumbai","BBQ Nation","Indian",1500,60,"Card","Delive
    ("O010","Anjali","Chennai","Burger Zone","American",480,32,"Cash","De
    ("O011","Farhan","Delhi","Biryani House","Indian",520,36,"UPI","Deliv
    ("O012","Megha","Hyderabad","Sushi Bar","Japanese",1100,58,"Card","Ca
    ("O013","Suresh","Bangalore","Curry Leaf","Indian",420,29,"UPI","Deli
    ("O014","Divya","Mumbai","Pizza Town","Italian",780,42,"Wallet","Deli
    ("O015","Nikhil","Delhi","Pasta Street","Italian",690,47,"UPI","Deliv
    ("O016","Kavya","Chennai","Dragon Bowl","Chinese",560,34,"UPI","Deliv
    ("O017","Rohit","Hyderabad","BBQ Nation","Indian",1400,62,"Card","Del
    ("O018","Simran","Bangalore","Burger Zone","American",510,31,"Cash","
    ("O019","Ayesha","Mumbai","Taco Bell","Mexican",570,35,"UPI","Deliver
    ("O020","Manish","Delhi","Curry Leaf","Indian",390,27,"Wallet","Deliv
    ("O021","Priya","Hyderabad","Pizza Town","Italian",720,41,"Card","Del
    ("O022","Yash","Chennai","Sushi Bar","Japanese",1150,57,"UPI","Delive
    ("O023","Naina","Bangalore","Pasta Street","Italian",680,44,"UPI","De
    ("O024","Sameer","Mumbai","Dragon Bowl","Chinese",610,39,"Wallet","De
    ("O025","Ritika","Delhi","Burger Zone","American",500,30,"Cash","Deli
    ("O026","Gopal","Hyderabad","Curry Leaf","Indian",410,28,"UPI","Deliv
    ("O027","Tina","Bangalore","Pizza Town","Italian",760,43,"Card","Deli
    ("O028","Irfan","Mumbai","BBQ Nation","Indian",1550,65,"Card","Delive
    ("O029","Sahil","Chennai","Taco Bell","Mexican",590,37,"UPI","Deliver
    ("O030","Lavanya","Delhi","Dragon Bowl","Chinese",630,40,"Wallet","De
    ("O031","Deepak","Hyderabad","Burger Zone","American",520,33,"Cash","
    ("O032","Shweta","Bangalore","Curry Leaf","Indian",450,31,"UPI","Deli
    ("O033","Aman","Mumbai","Pizza Town","Italian",810,46,"Card","Deliver
    ("O034","Rekha","Chennai","Pasta Street","Italian",700,45,"UPI","Deli
    ("O035","Zubin","Delhi","BBQ Nation","Indian",1480,63,"Card","Deliver
    ("O036","Pallavi","Hyderabad","Dragon Bowl","Chinese",580,36,"Wallet"
    ("O037","Naveen","Bangalore","Taco Bell","Mexican",560,34,"UPI","Deli
    ("O038","Sonia","Mumbai","Sushi Bar","Japanese",1180,59,"Card","Deliv
    ("O039","Harish","Chennai","Burger Zone","American",490,29,"Cash","De
    ("O040","Kriti","Delhi","Curry Leaf","Indian",420,26,"UPI","Delivered
]

columns = [
    "order_id","customer_name","city","restaurant","cuisine",
    "order_amount","delivery_time_minutes","payment_mode","order_status"
]

df = spark.createDataFrame(data, columns)
df.show()
df.printSchema()
```

EXERCISES — MEDIUM LEVEL

CSV, JSON, PARQUET (Food Delivery Use Case)

SECTION A — CSV

Exercise 1

Write the full dataset to CSV with header enabled.

Output:

```
orders_csv/
```

Exercise 2

Read the CSV back and filter:

- order_amount > 700

Exercise 3

From CSV, show only:

- order_id
- city
- cuisine
- order_amount

Exercise 4

Sort orders by delivery_time_minutes descending and write result to CSV.

SECTION B — JSON

Exercise 5

Write only “Delivered” orders to JSON.

Output:

```
delivered_orders_json/
```

Exercise 6

Read JSON and filter:

- city = "Mumbai"
- payment_mode = "Card"

Exercise 7

Add a column:

```
delivery_category
```

Logic:

- delivery_time_minutes > 45 → "Late"
- else → "OnTime"

Write output to JSON.

Exercise 8

Force JSON output to a single partition and observe number of files created.

SECTION C — PARQUET

Exercise 9

Convert full dataset to Parquet.

Output:

```
orders_parquet/
```

Exercise 10

Read Parquet and filter:

- cuisine = "Indian"
- order_amount > 500

Exercise 11

Sort Parquet data by order_amount descending and write top 10 orders back to Parquet.

Exercise 12

Compare storage size of:

- CSV
- JSON
- Parquet

Answer:

- Which is smallest?
- Why?

SECTION D — FORMAT CONVERSION

Exercise 13

Convert:

- CSV → Parquet
- JSON → Parquet

Exercise 14

Read Parquet and write it back as CSV using delimiter `|` .

THINKING / ANALYTICS QUESTIONS

Exercise 15

Which cuisine generates the highest order_amount overall?

Exercise 16

Which city has the highest number of orders?

Exercise 17

Which payment mode is most frequently used?

Exercise 18

Why is Parquet the preferred format for analytics platforms like Databricks and BigQuery?

OPTIONAL CHALLENGE

Challenge 1

Repartition the dataset into 4 partitions and write to Parquet.

Challenge 2

Create a report dataset containing:

- city
- total_orders
- total_revenue

Write it to Parquet.
