USE CASE

Ride-Sharing Trips Analytics (Uber / Ola-style)

This prepares cohorts for aggregations, performance analysis, and later window functions.

DATASET: Ride Trips (45 Records)

Schema

- trip_id
- rider_name
- city
- driver_name
- vehicle_type
- distance_km
- trip_fare
- trip_duration_minutes
- payment_mode
- trip_status

Create the Dataset in PySpark

```
data = [
    ("T001","Amit","Hyderabad","Ramesh","Sedan",12.5,320,28,"UPI","Comple
    ("T002","Neha","Bangalore","Suresh","Mini",8.2,210,22,"Card","Complet
    ("T003","Rahul","Delhi","Anil","Bike",5.1,120,15,"Cash","Completed"),
    ("T004","Pooja","Mumbai","Vikas","SUV",18.0,560,45,"UPI","Cancelled")
    ("T005","Arjun","Chennai","Kumar","Mini",7.8,200,20,"UPI","Completed"
    ("T006","Sneha","Hyderabad","Ramesh","Sedan",14.2,360,32,"Card","Comp
    ("T007","Karan","Delhi","Anil","Bike",6.3,140,18,"UPI","Completed"),
    ("T008","Riya","Bangalore","Suresh","Sedan",11.0,300,27,"Wallet","Com
    ("T009","Vikas","Mumbai","Vijay","SUV",20.5,650,50,"Card","Completed"
    ("T010","Anjali","Chennai","Kumar","Bike",4.9,110,14,"Cash","Complete
    ("T011","Farhan","Delhi","Anil","Mini",9.6,240,25,"UPI","Completed"),
    ("T012","Megha","Hyderabad","Ramesh","SUV",19.2,610,48,"Card","Cancel
    ("T013","Suresh","Bangalore","Suresh","Sedan",13.0,340,30,"UPI","Comp
    ("T014","Divya","Mumbai","Vikas","Mini",10.2,260,26,"Wallet","Complet
    ("T015","Nikhil","Delhi","Anil","Sedan",15.5,390,34,"UPI","Completed"
    ("T016","Kavya","Chennai","Kumar","Sedan",12.1,315,29,"UPI","Complete
    ("T017","Rohit","Hyderabad","Ramesh","SUV",22.0,700,55,"Card","Comple
    ("T018","Simran","Bangalore","Suresh","Bike",5.8,130,16,"Cash","Compl
    ("T019","Ayesha","Mumbai","Vijay","Mini",9.9,250,24,"UPI","Completed"
    ("T020","Manish","Delhi","Anil","Bike",6.0,135,17,"Wallet","Completed
    ("T021","Priya","Hyderabad","Ramesh","Sedan",14.8,380,33,"Card","Comp
    ("T022","Yash","Chennai","Kumar","SUV",21.3,680,52,"UPI","Completed")
    ("T023","Naina","Bangalore","Suresh","Mini",10.7,270,28,"UPI","Comple
    ("T024","Sameer","Mumbai","Vikas","Sedan",13.9,350,31,"Wallet","Compl
    ("T025","Ritika","Delhi","Anil","Bike",5.4,125,16,"Cash","Completed")
    ("T026","Gopal","Hyderabad","Ramesh","Mini",8.9,225,23,"UPI","Complet
    ("T027","Tina","Bangalore","Suresh","Sedan",12.6,330,29,"Card","Compl
    ("T028","Irfan","Mumbai","Vijay","SUV",23.4,740,58,"Card","Completed"
    ("T029","Sahil","Chennai","Kumar","Mini",9.4,235,24,"UPI","Completed"
    ("T030","Lavanya","Delhi","Anil","Sedan",14.1,365,32,"Wallet","Comple
    ("T031","Deepak","Hyderabad","Ramesh","Bike",6.7,150,18,"Cash","Compl
    ("T032","Shweta","Bangalore","Suresh","Mini",10.0,255,26,"UPI","Compl
    ("T033","Aman","Mumbai","Vikas","Sedan",15.8,395,35,"Card","Completed
    ("T034","Rekha","Chennai","Kumar","Sedan",13.5,345,30,"UPI","Complete
    ("T035","Zubin","Delhi","Anil","SUV",24.0,760,60,"Card","Completed"),
    ("T036","Pallavi","Hyderabad","Ramesh","Mini",9.1,230,23,"Wallet","Co
    ("T037","Naveen","Bangalore","Suresh","Bike",5.9,135,17,"UPI","Comple
    ("T038","Sonia","Mumbai","Vijay","SUV",21.7,690,54,"Card","Completed"
    ("T039","Harish","Chennai","Kumar","Mini",8.5,215,21,"Cash","Complete
    ("T040","Kriti","Delhi","Anil","Sedan",14.6,375,33,"UPI","Completed")
    ("T041","Apoorva","Hyderabad","Ramesh","Sedan",13.2,335,30,"Card","Co
    ("T042","Mohit","Bangalore","Suresh","SUV",19.9,620,49,"UPI","Complet
    ("T043","Tanvi","Mumbai","Vikas","Mini",10.4,265,27,"Wallet","Complet
    ("T044","Rakesh","Chennai","Kumar","Bike",6.2,140,18,"Cash","Complete
    ("T045","Isha","Delhi","Anil","Mini",9.7,245,25,"UPI","Completed")
]

columns = [
    "trip_id","rider_name","city","driver_name","vehicle_type",
    "distance_km","trip_fare","trip_duration_minutes",
    "payment_mode","trip_status"
]
```

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

EXERCISES — MEDIUM LEVEL

CSV, JSON, PARQUET (Ride-Sharing Use Case)

SECTION A — CSV

Exercise 1

Write the full dataset to CSV with header enabled.

Output:

```
trips_csv/
```

Exercise 2

Read the CSV and filter:

- trip_fare > 400
- trip_status = "Completed"

Exercise 3

From CSV, select:

- trip_id
- city
- vehicle_type
- trip_fare

Sort by trip_fare descending.

Exercise 4

Write only Bike trips to CSV using delimiter `|`.

SECTION B — JSON

Exercise 5

Write only trips from Mumbai to JSON.

Output:

```
mumbai_trips_json/
```

Exercise 6

Read JSON and add a column:

```
fare_per_km = trip_fare / distance_km
```

Write back to JSON.

Exercise 7

Filter JSON data:

- payment_mode = "Card"
- vehicle_type = "SUV"

Exercise 8

Force JSON output into a single partition and observe the output structure.

SECTION C — PARQUET

Exercise 9

Convert full dataset to Parquet.

Output:

```
trips_parquet/
```

Exercise 10

Read Parquet and filter:

- trip_duration_minutes > 45

Exercise 11

Sort Parquet data by distance_km descending and write top 10 trips back to Parquet.

Exercise 12

Compare storage size of:

- CSV
- JSON
- Parquet

Answer which is smallest and why.

SECTION D — FORMAT CONVERSION

Exercise 13

Convert:

- CSV → Parquet
- JSON → Parquet

Exercise 14

Read Parquet and write it back as CSV with header and delimiter `,` .

ANALYTICS THINKING QUESTIONS

Exercise 15

Which city generates the highest total trip_fare?

Exercise 16

Which vehicle_type has the highest average fare?

Exercise 17

Which driver has completed the most trips?

Exercise 18

Why is Parquet preferred for analytics dashboards and aggregations?

OPTIONAL CHALLENGE

Challenge 1

Repartition the dataset into 4 partitions and write to Parquet.

Challenge 2

Create a summary dataset with:

- city
- total_trips
- total_revenue

- average_trip_duration

Write it to Parquet.
