**Batch vs Real-Time Processing**

**Step 1. Create Lab Files**

**1.1 Generate batch\_orders.csv**

Open **Jupyter Notebook** or **Python script** and run:

import pandas as pd

import random

from datetime import datetime, timedelta

start\_time = datetime(2025, 9, 1, 9, 0, 0)

customers = ["Alice","Bob","Charlie","David","Emma","Frank","Grace","Hannah","Ivy","Jack","Zara"]

orders = []

for i in range(1, 101):

ts = start\_time + timedelta(minutes=i\*5)

cust = random.choice(customers)

amt = random.randint(50, 500)

orders.append([i, ts, cust, amt])

df = pd.DataFrame(orders, columns=["order\_id","timestamp","customer","amount"])

df.to\_csv("batch\_orders.csv", index=False)

print("batch\_orders.csv created")

This will create a file batch\_orders.csv in your current folder.

**1.2 Generate stream\_orders.txt**

Save this script as stream\_generator.py and run it in a **separate terminal (Command Prompt or PowerShell)**:

import time, random

from datetime import datetime

customers = ["Alice","Bob","Charlie","David","Emma","Frank","Grace","Hannah","Ivy","Jack","Zara"]

order\_id = 101

with open("stream\_orders.txt", "a") as f:

while True:

ts = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

cust = random.choice(customers)

amt = random.randint(50, 500)

line = f"{order\_id},{ts},{cust},{amt}\n"

f.write(line)

f.flush()

print("New order:", line.strip())

order\_id += 1

time.sleep(5)

* This will **append a new order every 5 seconds** to stream\_orders.txt.
* Keep this running in one terminal while you analyze in another.

**Step 2. Batch Processing (Jupyter Notebook)**

import pandas as pd

# Read batch file at once

batch\_df = pd.read\_csv("batch\_orders.csv")

print("First 5 rows of batch data:")

print(batch\_df.head())

# Simple analytics

print("\nBatch total revenue:", batch\_df['amount'].sum())

print("Batch average order value:", batch\_df['amount'].mean())

print("Orders per customer:")

print(batch\_df['customer'].value\_counts())

**Step 3. Real-Time Processing on Windows**

Since Windows doesn’t have tail -f natively, use **PowerShell**:

Get-Content .\stream\_orders.txt -Wait

This will continuously print new orders as they are written.

**Real-Time Analytics in Python**

import time

def read\_stream(filename="stream\_orders.txt"):

with open(filename, "r") as f:

# move to end of file

f.seek(0,2)

while True:

line = f.readline()

if not line:

time.sleep(1) # wait for new data

continue

yield line.strip()

print("Listening to stream...")

for order in read\_stream():

order\_id, ts, cust, amt = order.split(",")

print(f"Real-time order: {cust} bought worth ${amt} at {ts}")

**Step 4. Comparison Notes**

**Batch Processing**

* **Pros:** Efficient for large historical datasets, cheaper infra.
* **Cons:** High latency (you wait until batch completes).

**Real-Time Processing**

* **Pros:** Low latency, immediate insights.
* **Cons:** Higher infra costs, more complex design.

**Conclusion**

* Batch = historical analytics.
* Streaming = instant reactions.
* In practice, both are combined in modern **Lambda/Kappa architectures**.