Assignment 2

1. Assignment Objective (20 Marks)

You are tasked with building a real-time streaming pipeline for a Food Delivery platform (like Zomato/Swiggy).

Goal:
Whenever a new food order is inserted into PostgreSQL, your program should:

1. Detect the new record
2. Publish it as a JSON event to a Kafka topic
3. Consume the event using Spark Structured Streaming
4. Process, clean, and store the order into a Data Lake (Parquet format)

2. PostgreSQL Input Table

Food Delivery Orders

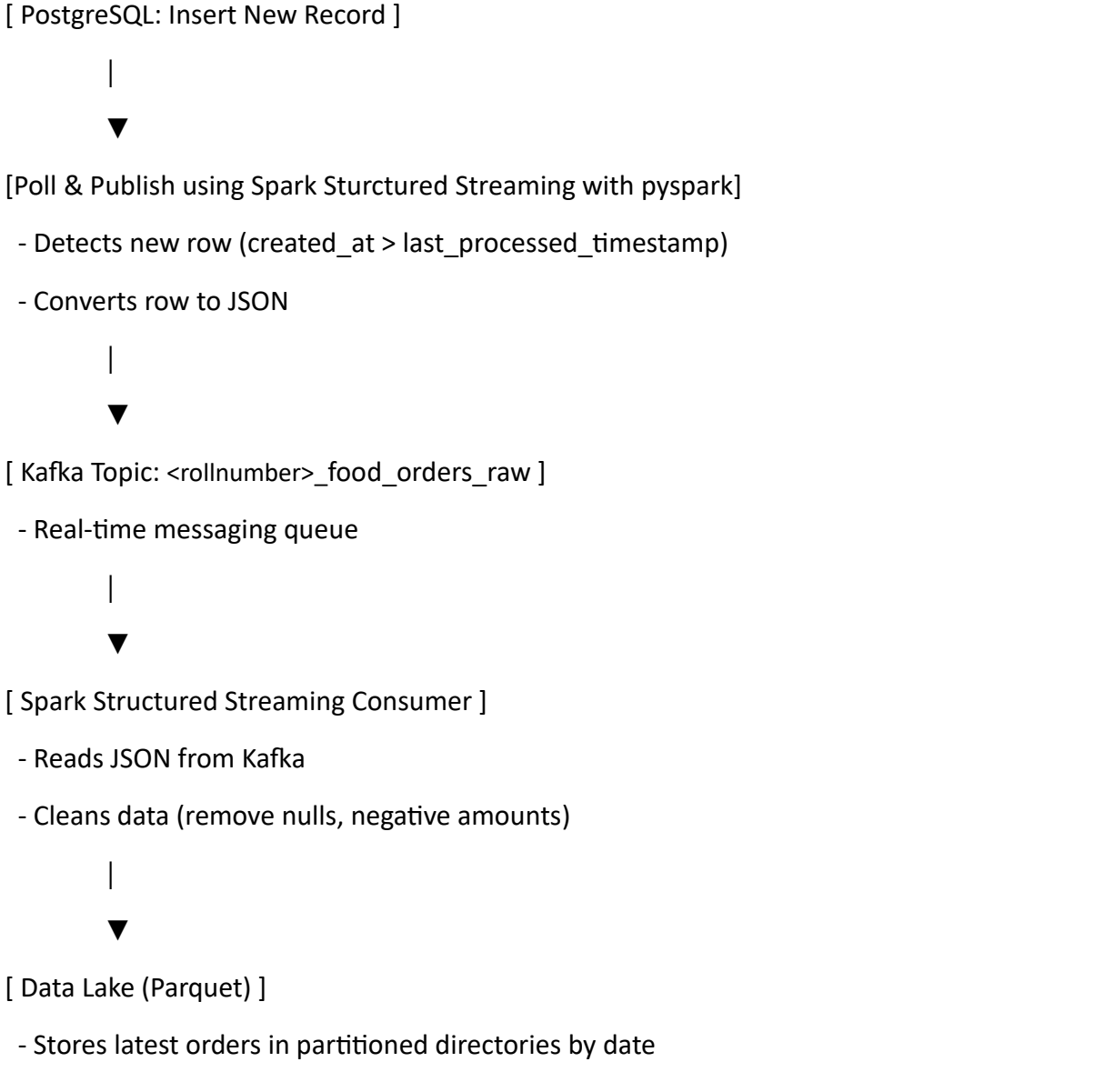
- Each order contains:

Column	Type	Description
order_id	INTEGER	Unique Order ID (SERIAL)
customer_name	VARCHAR	Name of the customer
restaurant_name	VARCHAR	Name of the restaurant
item	VARCHAR	Ordered item
amount	NUMERIC	Price of the item
order_status	VARCHAR	PLACED, PREPARING, DELIVERED, CANCELLED
created_at	TIMESTAMP	Record creation timestamp

- All new rows inserted into PostgreSQL must flow in real-time into the Data Lake via Kafka + Spark Streaming.

3. Architecture

This pipeline demonstrates the classic data evolution pattern:



Key Points:

1. **Incremental ingestion**: Only new rows are published to Kafka
2. **Streaming processing**: Spark consumes continuously
3. **Data Lake storage:** Append-only, partitioned by created_at date
4. New records → Kafka → Process → Persist in Lake → Ready for analytics
5. Students must use a **standard config file (orders_stream.yml)** for all pipeline parameters

4. Assignment Tasks

Part 1 — PostgreSQL Setup

1. Create table orders as described above.
2. Insert at least 10 initial sample records.
3. Verify that created_at is correctly populated.
4. Later, you will ingest **5 incremental records** to test incremental ingestion.

Deliverable: db/orders.sql (which has create and insert related scripts)

Part 2 — CDC Simulation & Kafka Producer using Pyspark

Deliverable: producers/orders_cdc_producer.py

Requirements:

1. Connect to PostgreSQL.
2. Poll the table every 5 seconds for new rows using created_at. (use created_at > last_processed_timestamp. Here you need to maintain last_processing_timestamp details in a separate file).
3. Convert new rows to JSON format like below:

```
{
 "order_id": 101,
 "customer_name": "John Doe",
 "restaurant_name": "Burger Junction",
 "item": "Veg Burger",
 "amount": 220,
 "order_status": "PLACED",
 "created_at": "2025-11-18T12:24:00Z"
}
```

4. Publish each JSON to Kafka topic: <rollnumber>_food_orders_raw
5. Maintain last processed timestamp to avoid duplicates.

Validation Rules:

- Must poll periodically and detect new inserts
- JSON must match schema exactly

- Correct Kafka topic name

Part 3 — Kafka → Spark Structured Streaming Consumer using pyspark

Deliverable: consumers/orders_stream_consumer.py

Requirements:

1. Consume <rollnumber>_food_orders_raw topic from Kafka.
2. Parse JSON into Spark DataFrame using correct schema.
3. Data Cleaning:
    - Remove records with null order_id or negative amount
4. Write to Data Lake path defined in config (s3://bucket/... or local path).
5. Partition data by **date (**YYYY-MM-DD**)**. Derive this date from created_at column

datalake/food/orders/

└── date=YYYY-MM-DD/

- Format: Parquet
- Mode: Append
- Use checkpointing to maintain streaming state (to store the offsets)

Validation Rules:

- Spark must run as continuous streaming job
- Checkpoint directory must exist
- Partitioned by date correctly

Part 4 — Incremental Testing & Evaluation Pattern

Instructor will evaluate as follows:

1. Insert 5 **new records** in PostgreSQL.
2. Run **producer and consumer scripts**.
3. Validate **counts and correctness** in the final Data Lake File.
4. Insert **5 additional incremental records**.
5. Validate **counts and correctness** in the final Data Lake File.

6. Repeat the process.
7. Evaluate if **incremental ingestion worked without duplicates**.

Assignment Deliverables:

Instructions:

1. Project Submission:

   You must upload your zipped folder following the project structure provided below.

2. Configuration File (orders_stream.yml):

   Your input file should contain only the variables listed in the sample configuration provided.

3. Spark Submit Command:

   Use the similar Spark submit command as shown in the sample (only one config file as an input)

4. Intermediate S3 file access:

   If you are storing intermediate data in S3 buckets, ensure that your buckets are set to global access, allowing any user to read and write. If you are using the local file system, you can ignore this instruction.

5. You can use Pyspark, Spark Structured Streaming, Kafka and PostgreSQL as a tech stack here.

1. Submission Structure:

Note: #if you are using s3 bucket, mention “s3” here otherwise you can use local files and mention “local”

<rollnumber>/food_delivery_streaming/<s3 or local>/

```
├── db/
|   └── orders.sql
├── producers/
|   └── orders_cdc_producer.py
├── consumers/
|   └── orders_stream_consumer.py
├── scripts/
|   └── producer_spark_submit.sh
|    └── consumer_spark_submit.sh
├── configs/
|   └── orders_stream.yml
└── README.md
```

Sample Config File:

File: configs/orders_stream.yml

Parameters to include:

Note: change the values according to your environment, but keep the variable names same

```
postgres:
  jdbc_url : "jdbc:postgresql://127.0.0.1:5432/food_delivery_db "
  host: 127.0.0.1
  port: 5432
  db: "food_delivery_db"
  user: " student"
  password: " student123"
  table: <rollnumber>_orders

kafka:
  brokers: "localhost:9092"
  topic: "<rollnumber>_food_orders_raw"

datalake:
  path: "/datalake/food/<rollnumber>/output/orders"   # or s3://bucket/path
  format: "parquet"

streaming:
  checkpoint_location: "/datalake/food/<rollnumber>/checkpoints/orders" # to store
offsets
  last_processed_timestamp_location: "/datalake/food/<rollnumber>/lastprocess/orders" #
to store last processed timestamp details
  batch_interval: 5  # seconds
```

Sample Spark Submit commands:

File: scripts/producer_spark_submit.sh

```
source de_env/bin/activate && spark-submit --packages org.apache.spark:spark-sql-kafka-
0-10_2.12:3.5.1  orders_cdc_producer.py --config configs/orders_stream.yml
```

File: scripts/consumer_spark_submit.sh

```
source de_env/bin/activate && spark-submit --packages org.apache.spark:spark-sql-kafka-
0-10_2.12:3.5.1  orders_stream_consumer.py --config configs/orders_stream.yml
```