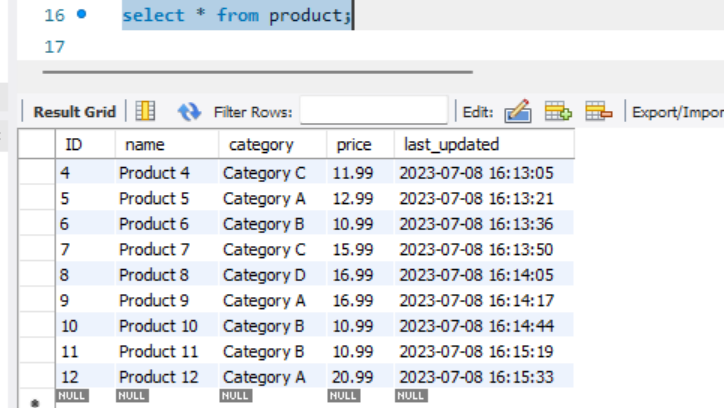
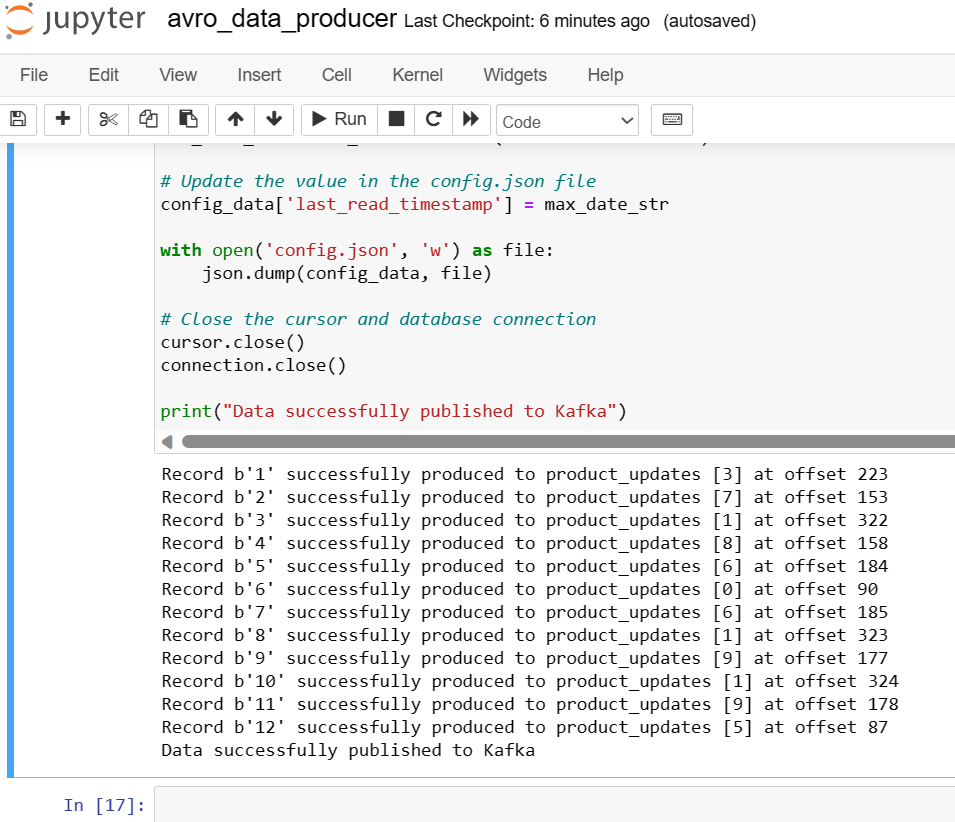
Orders Data Pub/Sub Using Kafka

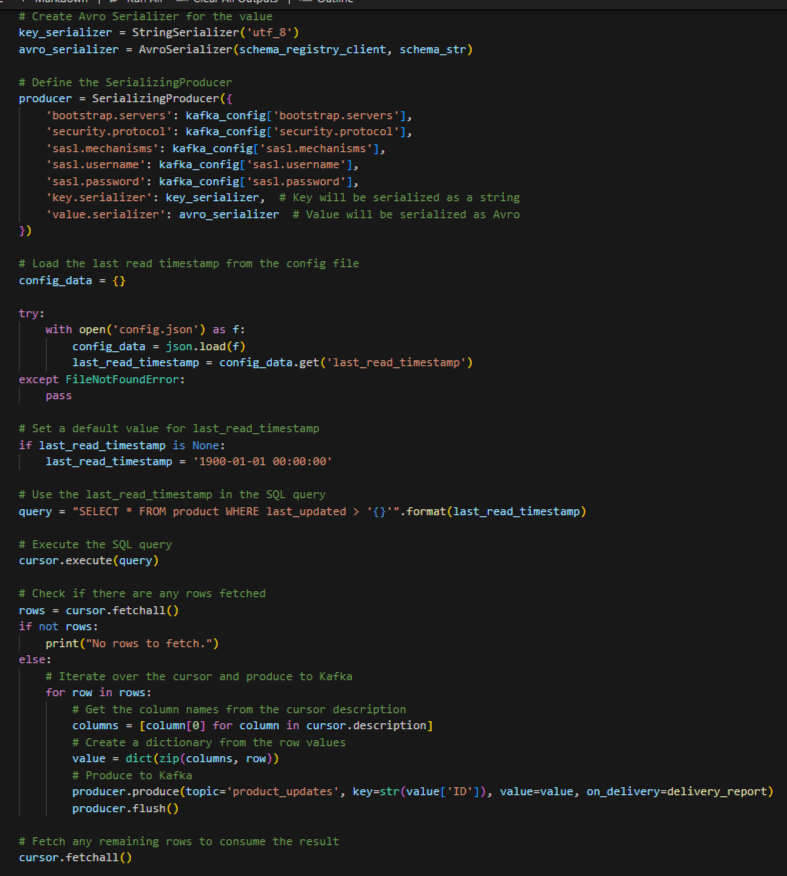
1. In this project, I builded a Kafka producer and a consumer group that worked with a MySQL database, Avro serialization, and multi-partition Kafka topics. The producer was fetching incremental data from a MySQL table and writing Avro serialized data into a Kafka topic. The consumers then deserializing this data and appending it to separate JSON files.
2. MySQL database storing product information such as product ID, name, category, price, and updated timestamp. Scenario is like I was updating the database frequently with new products and changing product information. I wanted to build a real-time system to stream these updates incrementally to a downstream system for real-time analytics.
3. Created a table named 'product' in MySQL database & initially loaded 12 records in the table ‘Product’ in MySQL database.



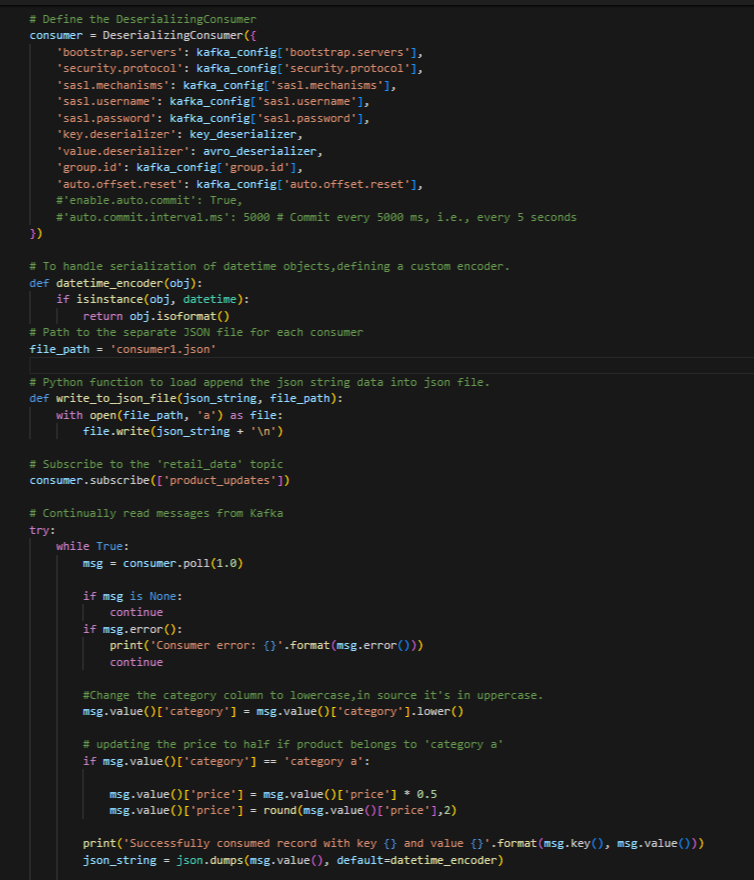
1. Wrote a Kafka producer in Python that used a MySQL connector to fetch data from the MySQL table. In my producer code, maintained a record of the last read timestamp. Each time I fetched data, used a SQL query to get records where the last\_updated timestamp is greater than the last read timestamp. Hence, records processed to producer successfully.



1. Serialized the data into Avro format and published the data to a Kafka topic named "product\_updates". Configured this topic with 10 partitions. Used the product ID as the key when producing messages. This ensured that all updates for the same product end up in the same partition.

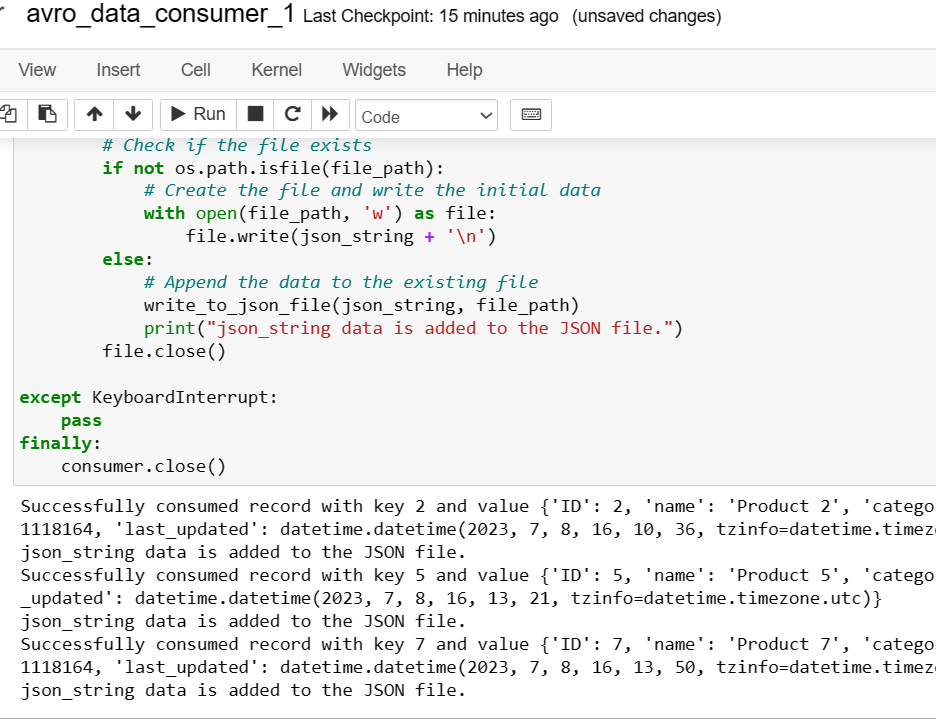


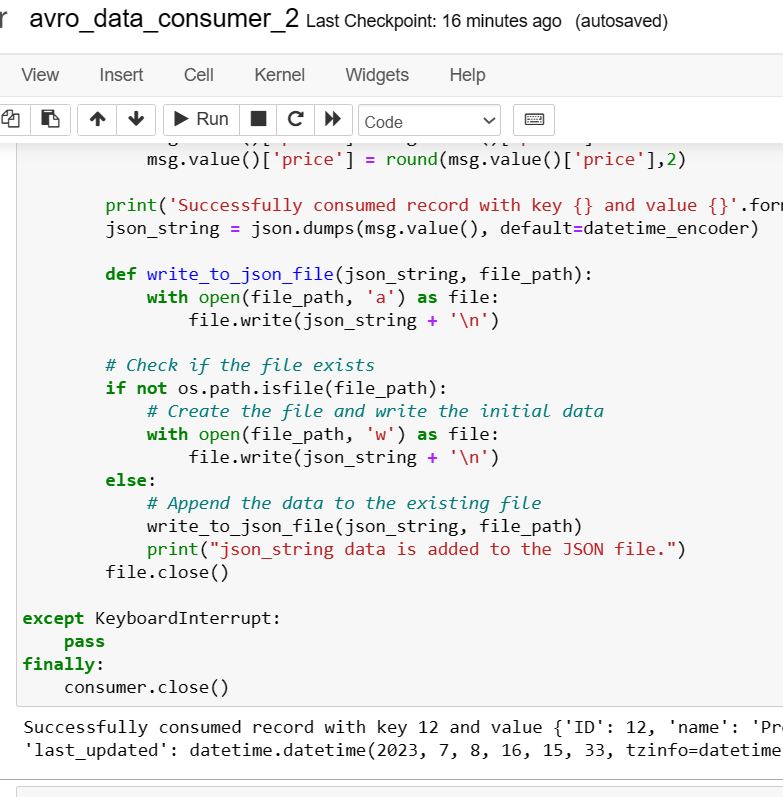
1. Wrote a Kafka consumer in Python and set it up as a consumer group of 5 consumers. Each consumer read data from the "product\_updates" topic. Deserialized the Avro data back into a Python object & implemented data transformation logic.

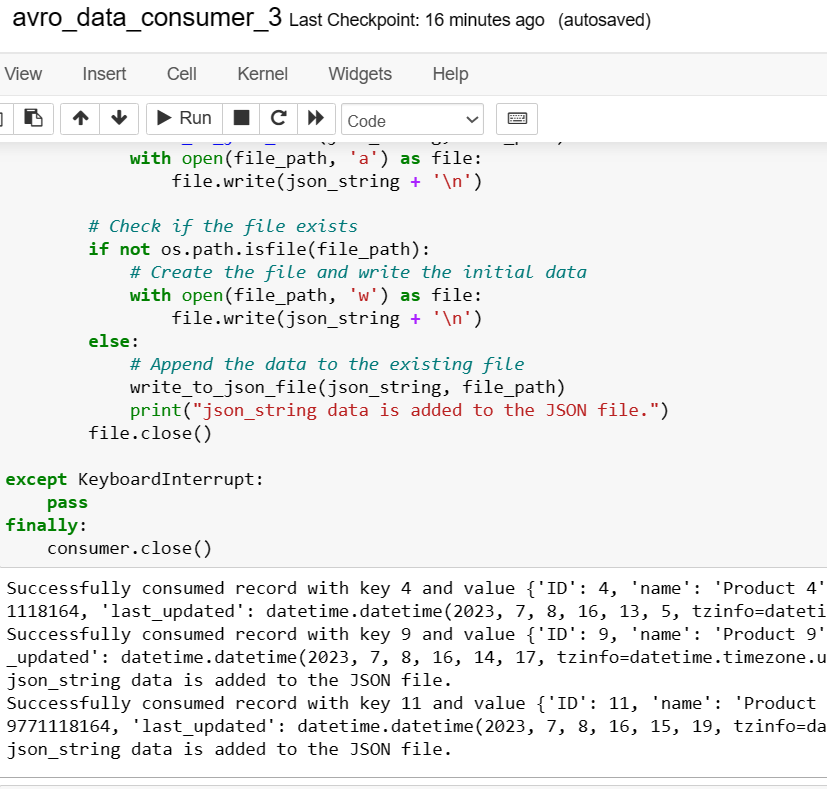


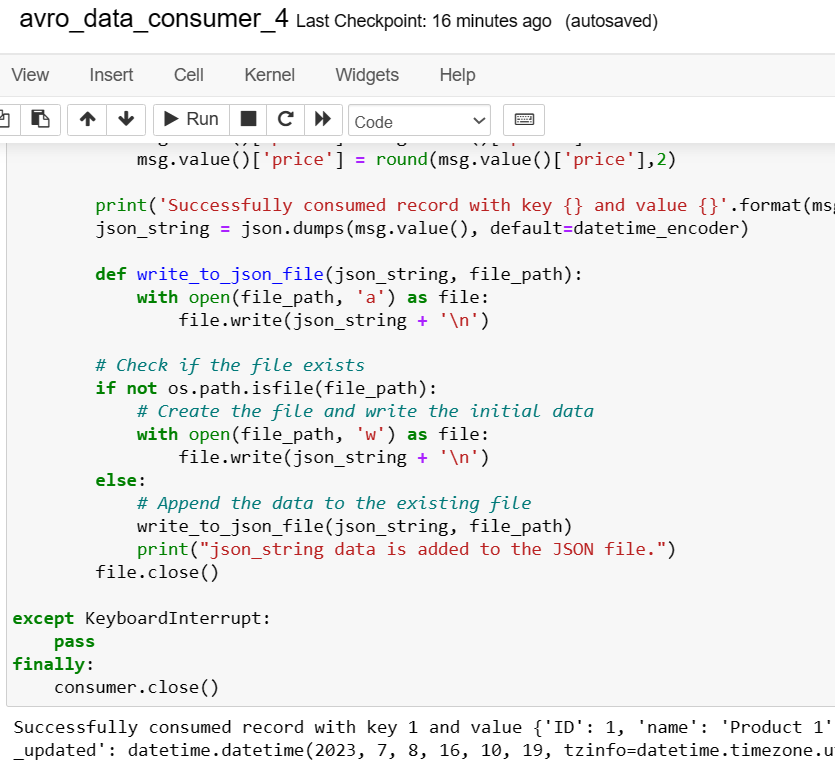
1. Consumers start consuming the 12 records( screenshots attached for consumer 1 to 5 consuming all 12 records in the table initially).

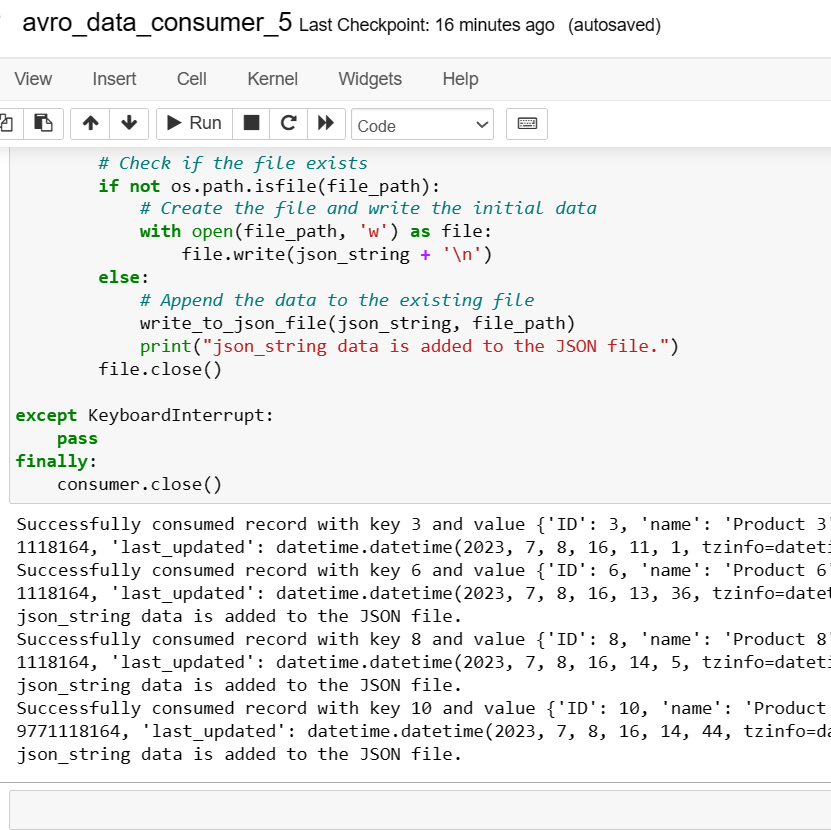
Each consumer converting the transformed Python object into a JSON string and appending the JSON string to a separate JSON file. Opened the file in append mode



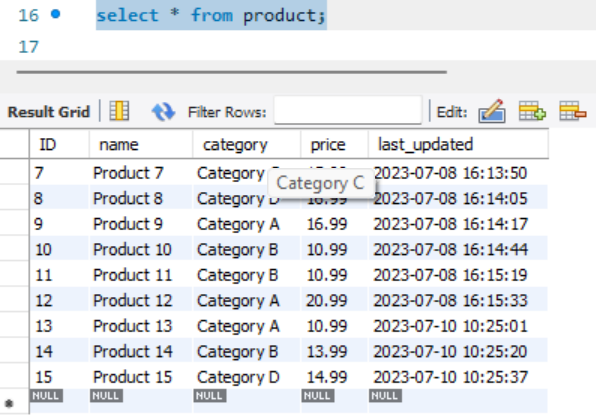




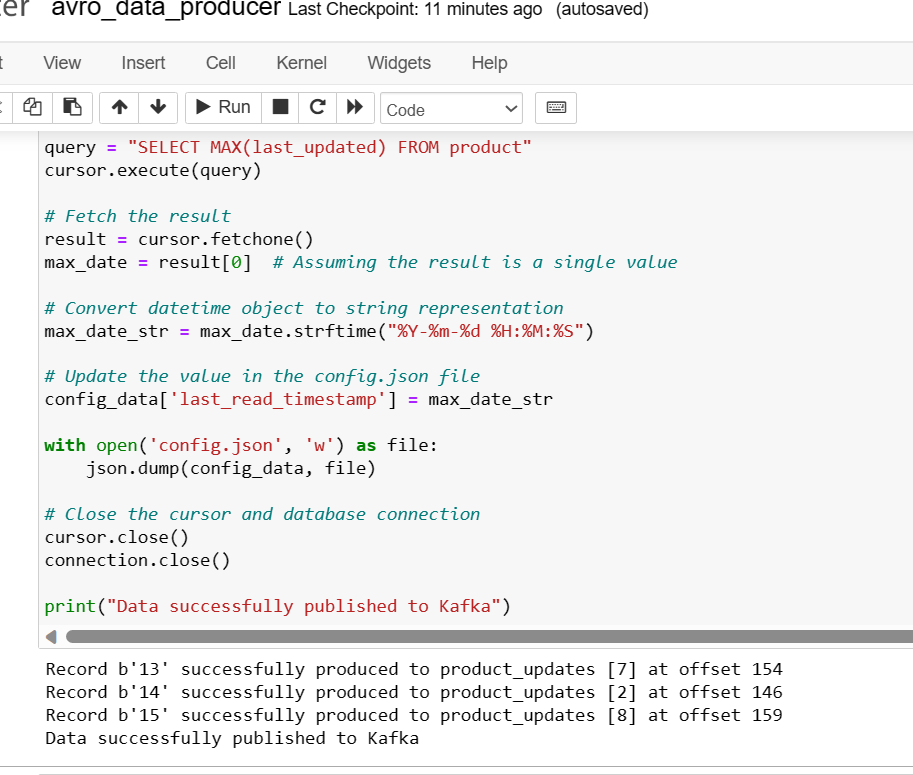




1. Again loaded 3 records in the table.



1. 3 new record processed to producer successfully.



1. All 3(with id as 13,14,15) records consumed by consumers 1,4 and 3 respectively.

