Laboratory 9

Lab 3

1. Create two topics:

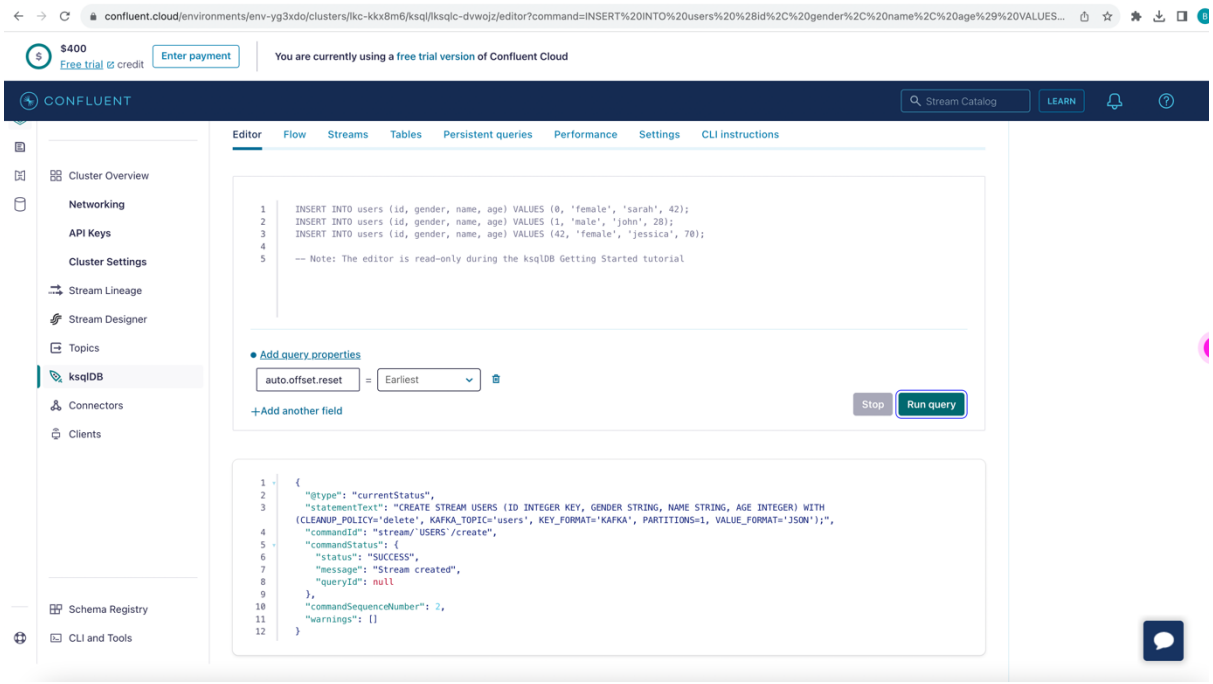
topic item in left part of menu, above ksqlDB.
Add topic --> ConfluentTextLinesTopic --> partitions 10
Add topic --> UppercasedTextLinesTopic --partitions 10

2. Produce some low-case values into ConfluentTextLinesTopic
topic item in left part of menu, above ksqlDB.
choose --> ConfluentTextLinesTopic --> Messages tab --> "+ Produce a new message to this topic"
"alpha" --> Produce
"beta" --> Produce

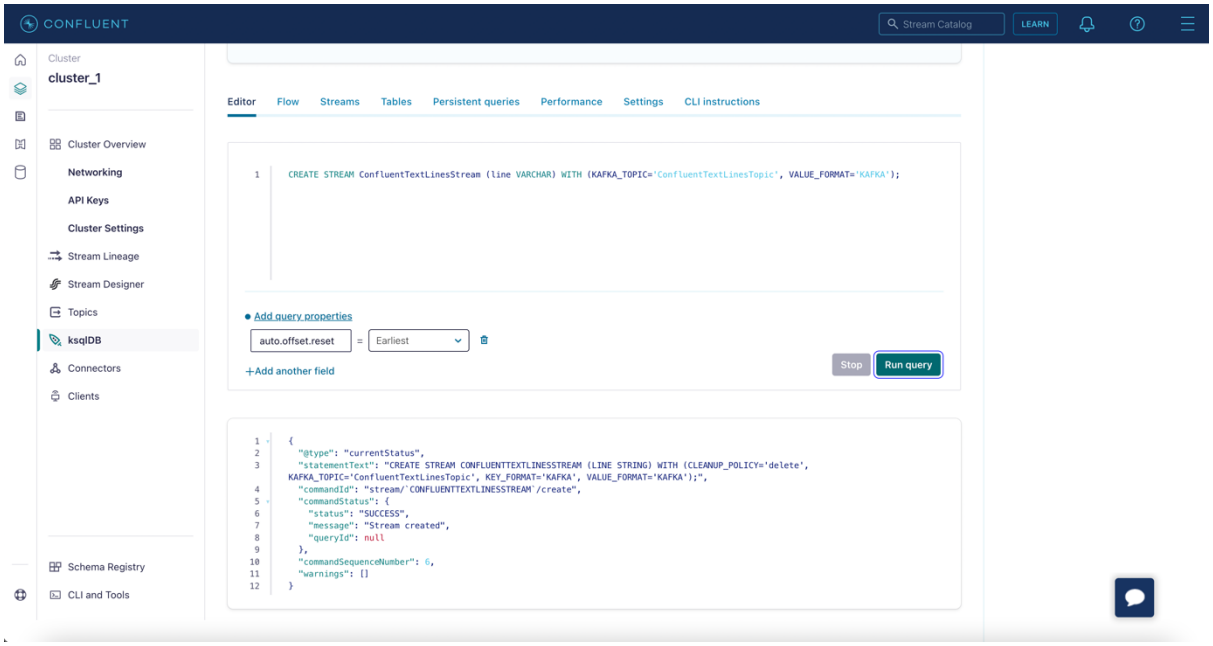
3. Run KSQL
ksqlDB item in left part of menu, under Topics.
ksqlDB_cluster_0 --> Editor
(or the name of cluster that you have created)

4. Run the commands inside KSQL

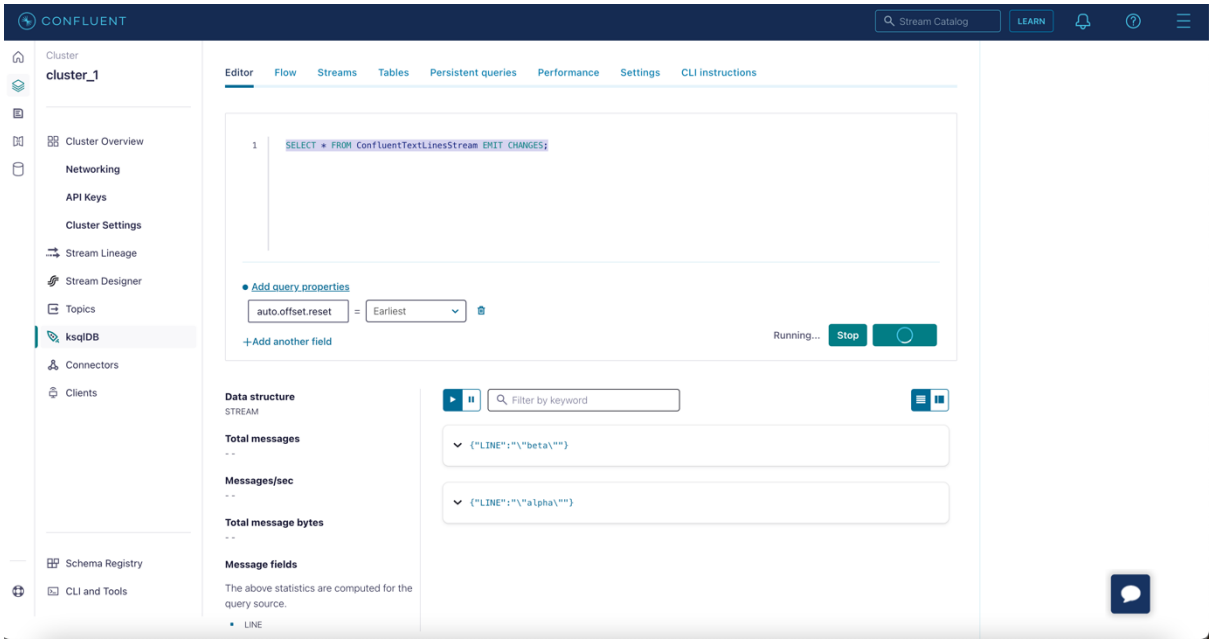
SET 'auto.offset.reset'='earliest';



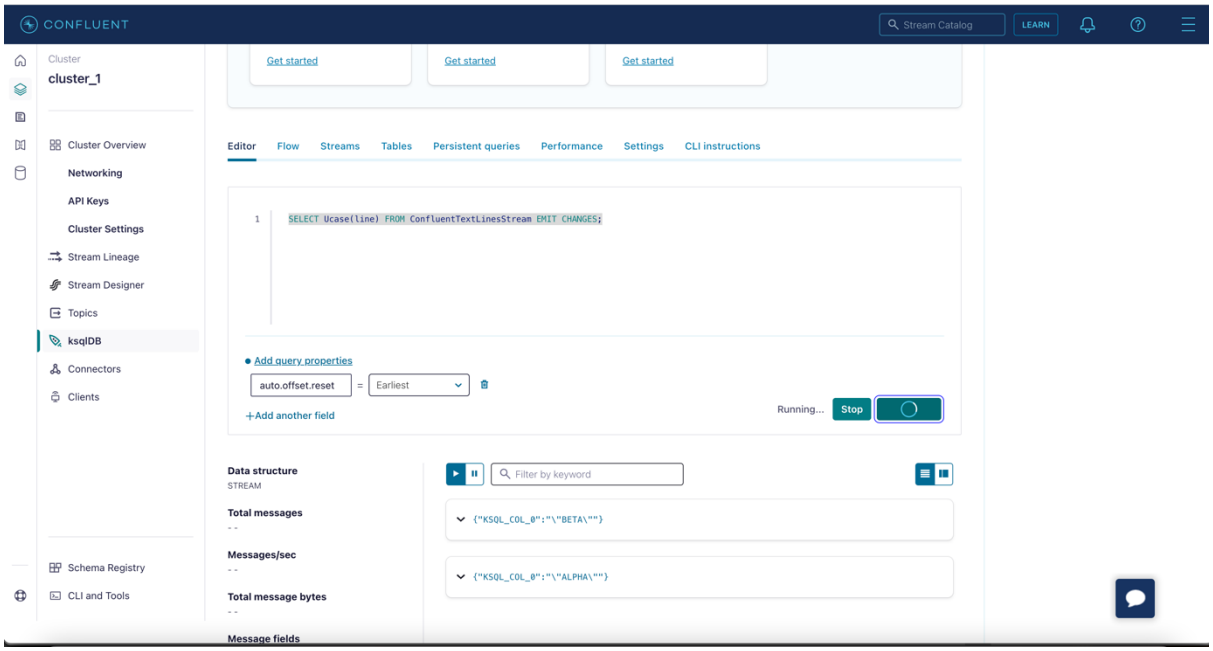
CREATE STREAM ConfluentTextLinesStream (line VARCHAR) WITH (KAFKA_TOPIC='ConfluentTextLinesTopic', VALUE_FORMAT='KAFKA');



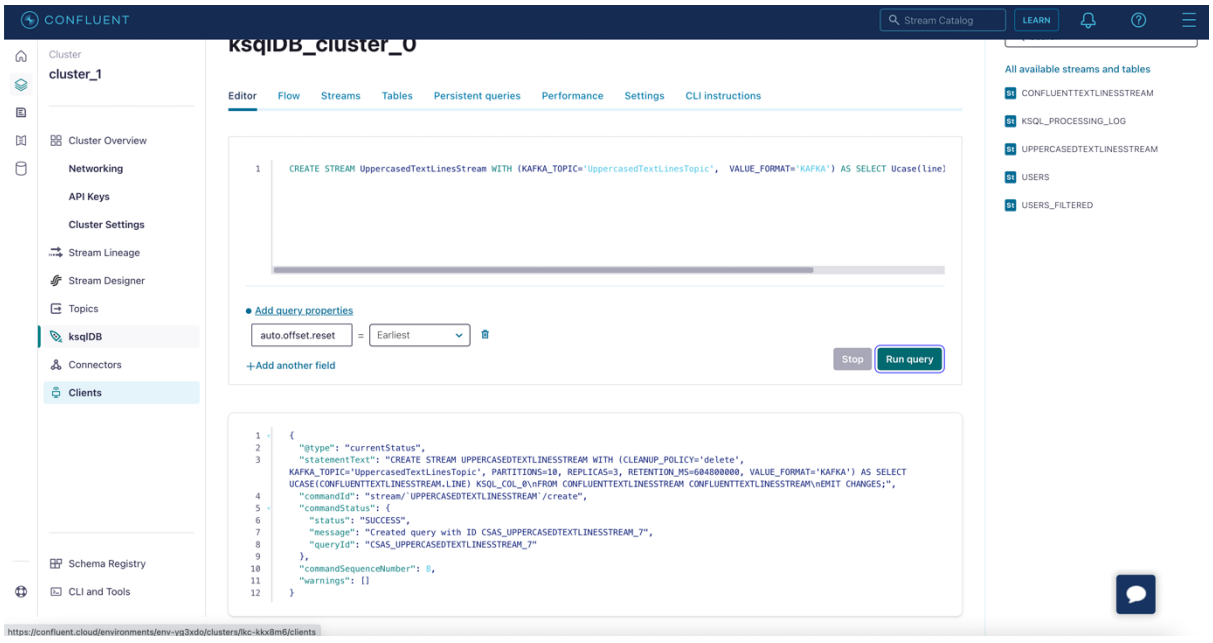
SELECT * FROM ConfluentTextLinesStream EMIT CHANGES;



SELECT Ucase(line) FROM ConfluentTextLinesStream EMIT CHANGES;

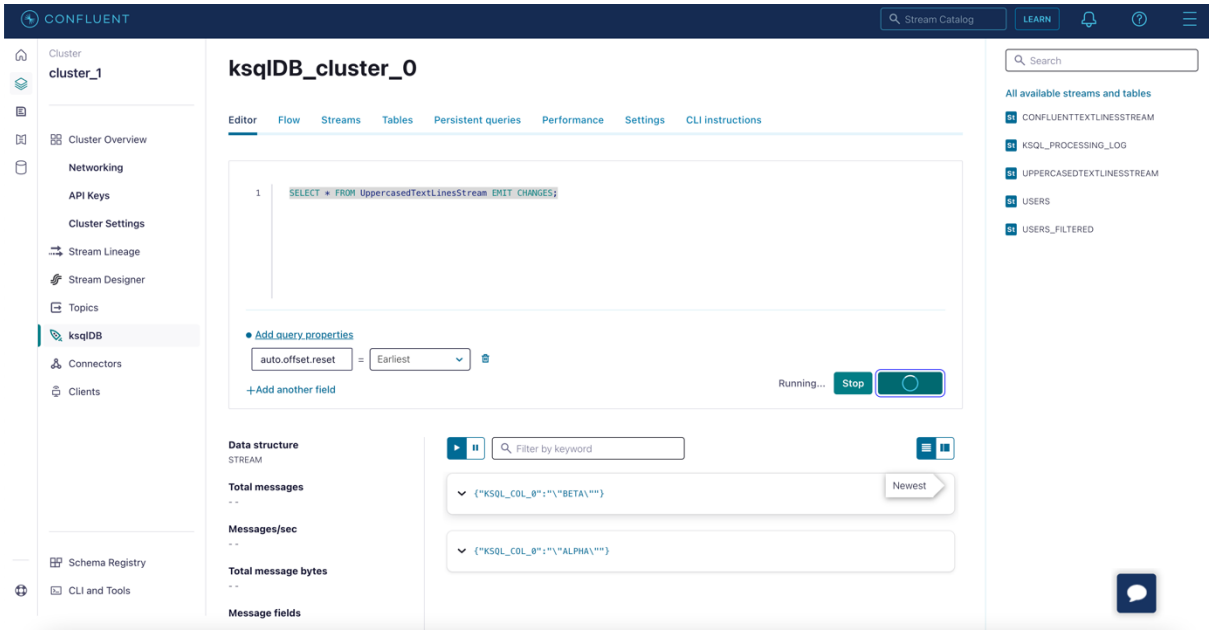


CREATE STREAM UppercasedTextLinesStream WITH (KAFKA_TOPIC='UppercasedTextLinesTopic', VALUE_FORMAT='KAFKA') AS SELECT Ucase(line) FROM ConfluentTextLinesStream;



5. Check UppercasedTextLinesTopic topic;

SELECT * FROM UppercasedTextLinesStream EMIT CHANGES;

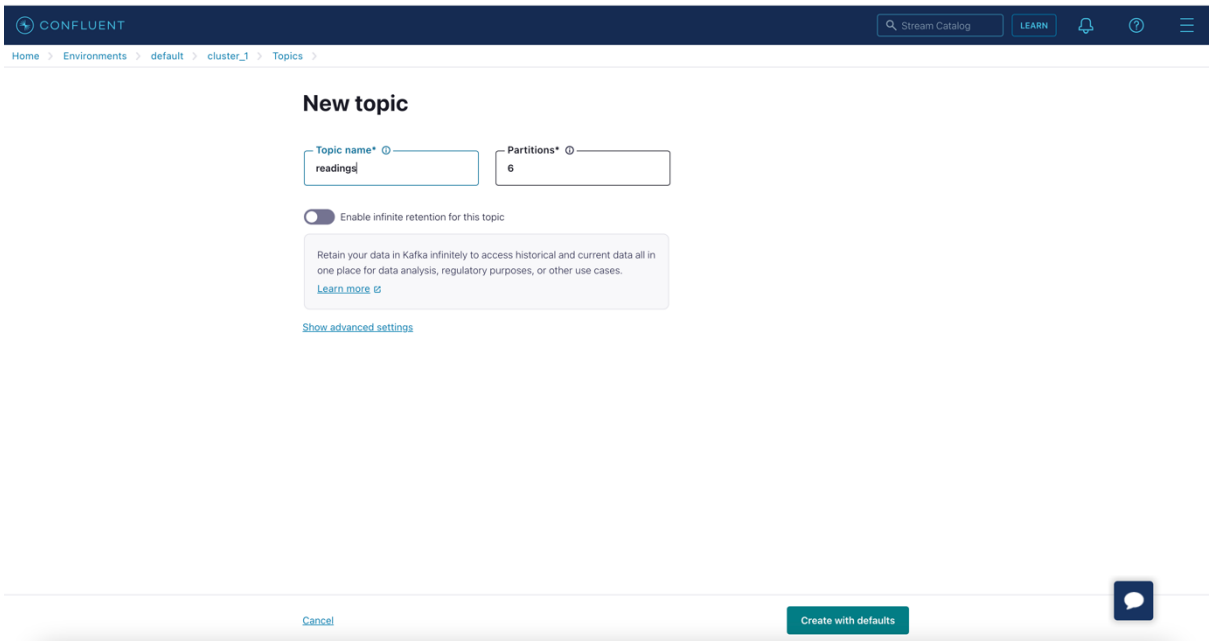


Topics:

Topic name	Partitions	Production (last min)	Consumption (last min)	Schema
ConfluentTextLinesTopic	10	0B/s	0B/s	Set a schema
pksqlc-11n23-processing-log	8	--	--	Set a schema
pksqlc-11n23USERS_FILTERED	1	0B/s	--	Set a schema
UppercasedTextLinesTopic	10	0B/s	0B/s	Set a schema
users	1	0B/s	0B/s	Set a schema

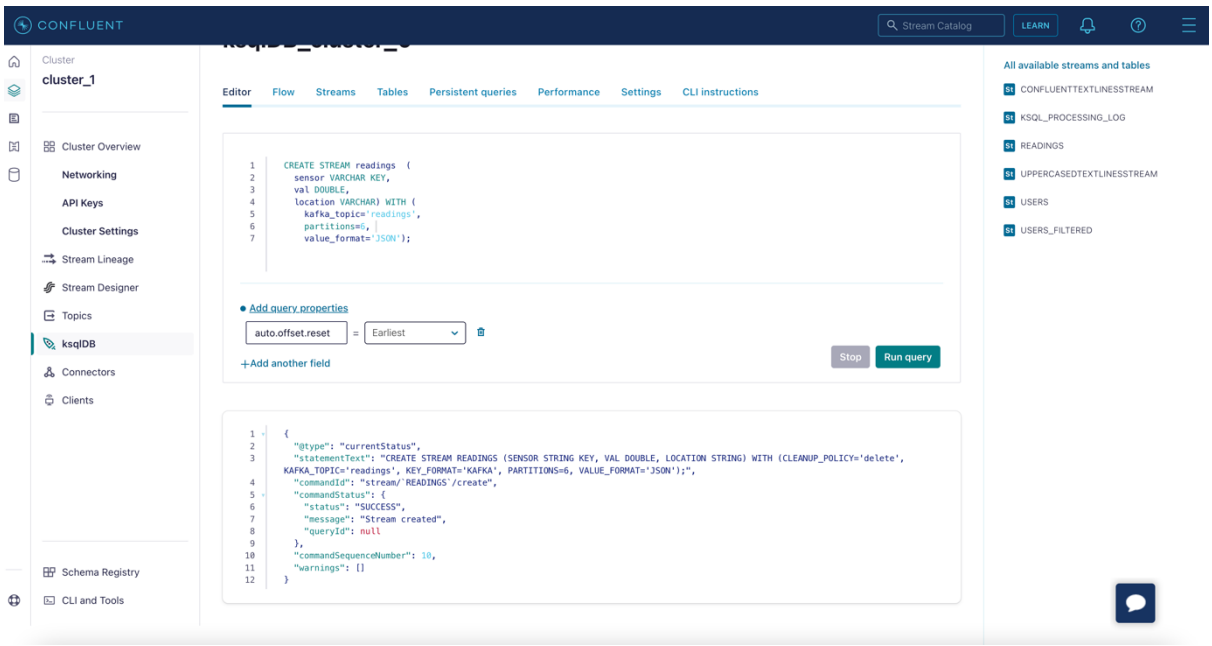
Lab 4

1. Create topic:
   readings --> partitions 6



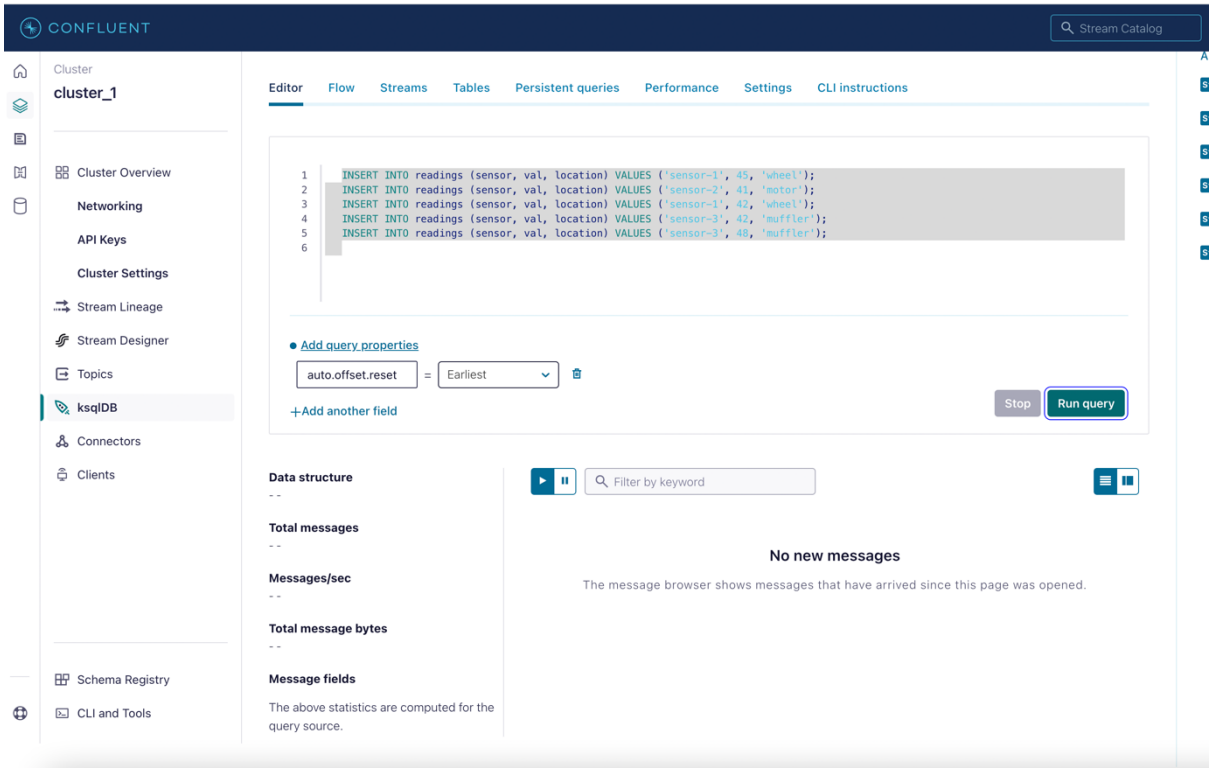
2. create stream for topic readings

```
CREATE STREAM readings  (
  sensor VARCHAR KEY,
  val DOUBLE,
  location VARCHAR) WITH (
    kafka_topic='readings',
    partitions=6,
    value_format='JSON');
```

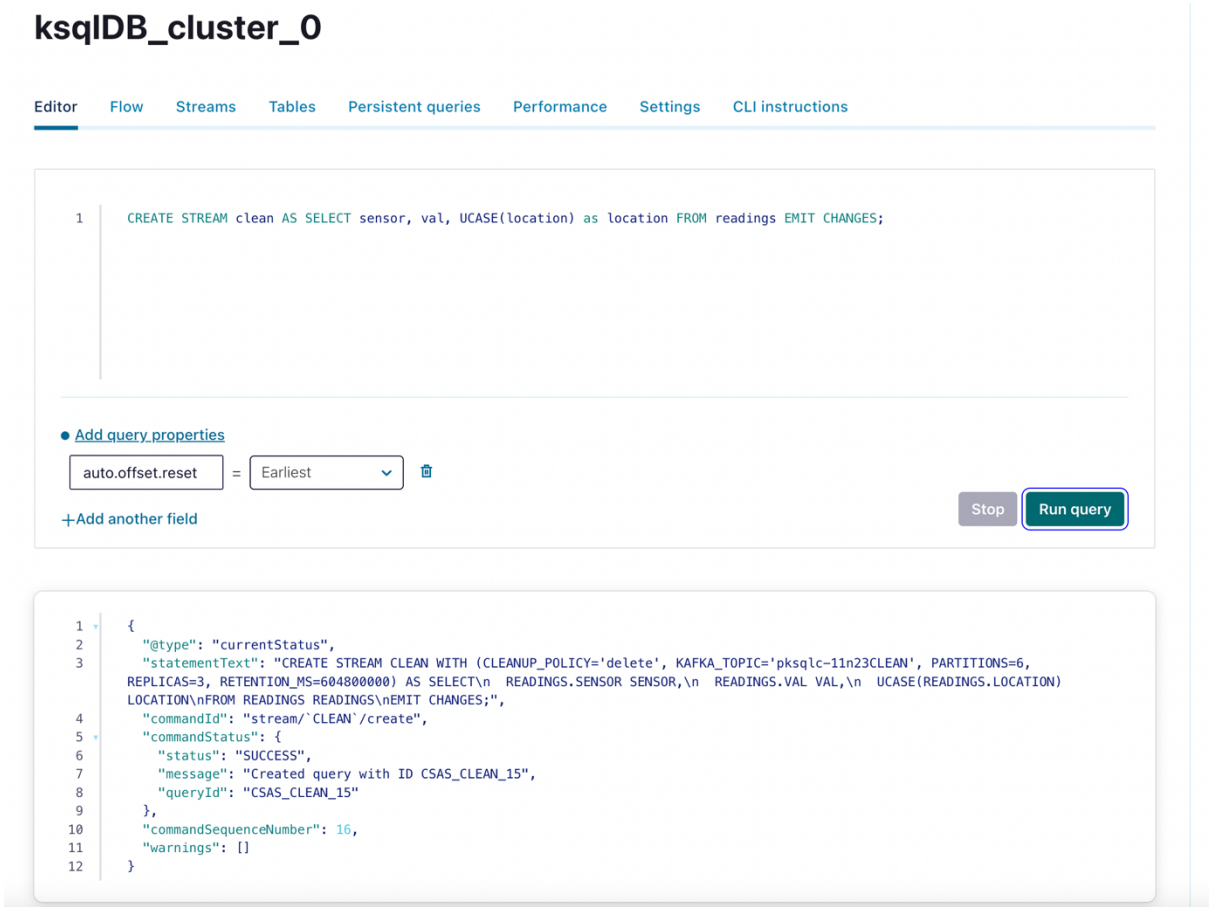


3. Produce some events into readings topic

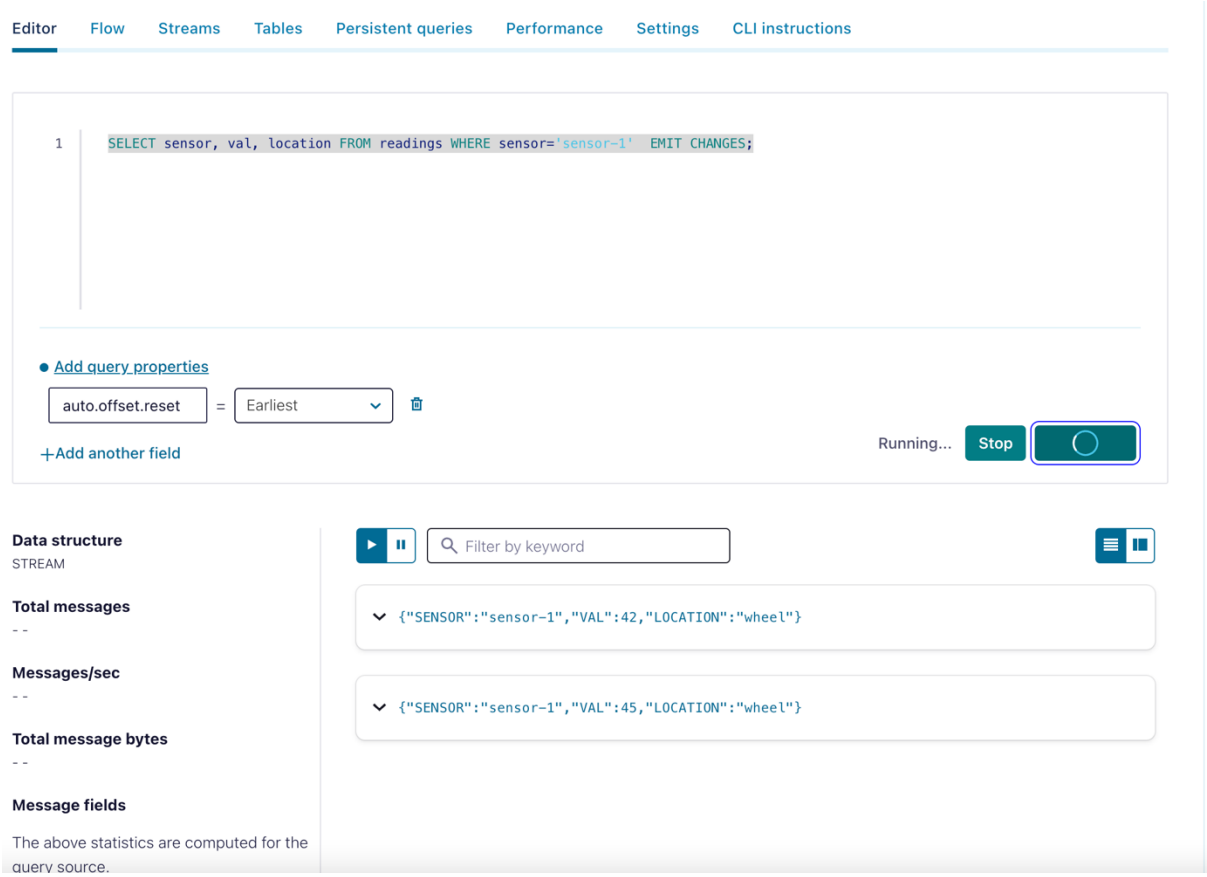
```
INSERT INTO readings (sensor, val, location) VALUES ('sensor-1', 45, 'wheel');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-2', 41, 'motor');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-1', 42, 'wheel');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-3', 42, 'muffler');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-3', 48, 'muffler');
```



4. Run some queries
CREATE STREAM clean AS SELECT sensor, val, UCASE(location) as location FROM readings EMIT CHANGES;



SELECT sensor, val, location FROM readings WHERE sensor='sensor-1' EMIT CHANGES;



CREATE STREAM high_readings AS SELECT sensor, val, location FROM clean where val > 42 EMIT CHANGES;

ksqlDB_cluster_0

Editor | Flow | Streams | Tables | Persistent queries | Performance | Settings | CLI instructions

```
CREATE STREAM high_readings AS SELECT sensor, val, location FROM clean where val > 42 EMIT CHANGES;
```

Add query properties

auto.offset.reset = Earliest

+Add another field

Stop | Run query

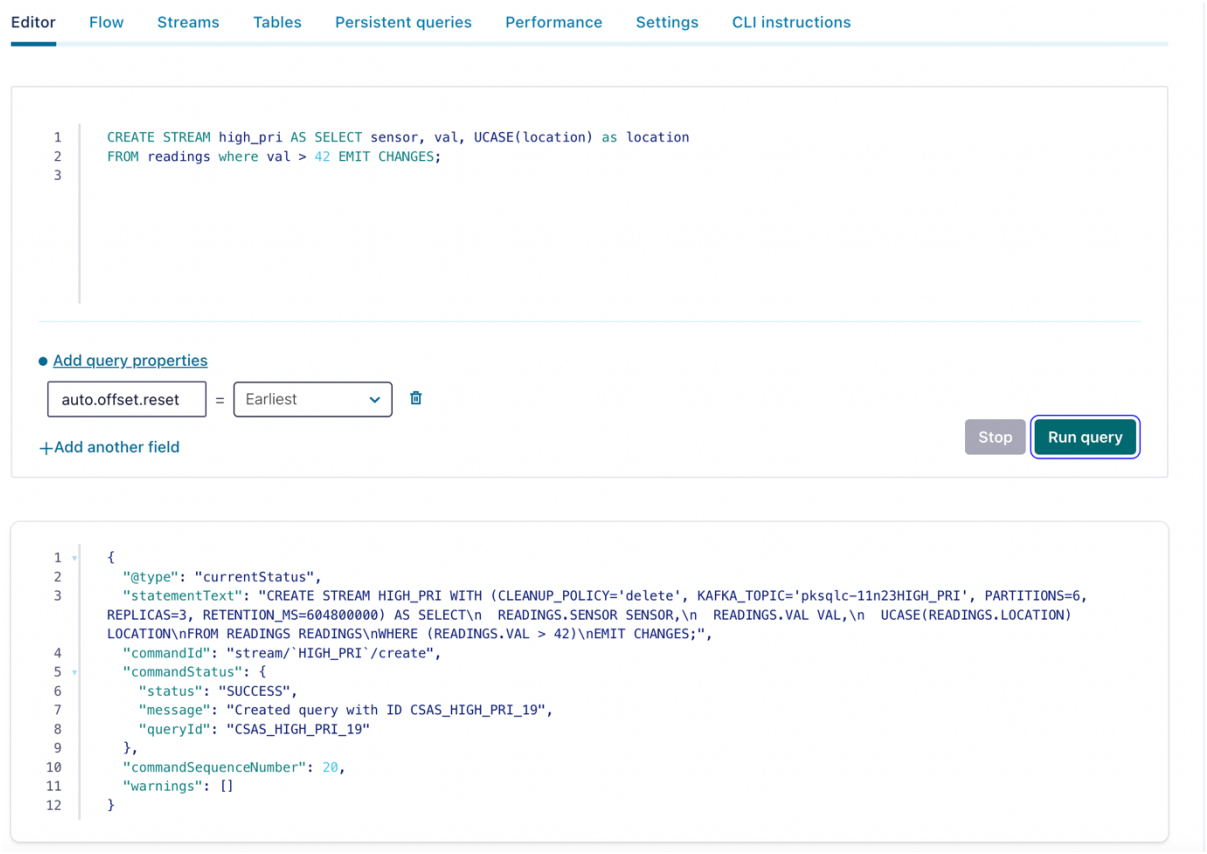

```
{
  "@type": "currentStatus",
  "statementText": "CREATE STREAM HIGH_READINGS WITH (CLEANUP_POLICY='delete', KAFKA_TOPIC='pksqlc-11n23HIGH_READINGS',
PARTITIONS=6, REPLICAS=3, RETENTION_MS=604800000) AS SELECT\n  CLEAN.SENSOR SENSOR,\n  CLEAN.VAL VAL,\n  CLEAN.LOCATION
LOCATION\nFROM CLEAN CLEAN\nWHERE (CLEAN.VAL > 42)\nEMIT CHANGES;",
  "commandId": "stream/`HIGH_READINGS`/create",
  "commandStatus": {
    "status": "SUCCESS",
    "message": "Created query with ID CSAS_HIGH_READINGS_17",
    "queryId": "CSAS_HIGH_READINGS_17"
  },
  "commandSequenceNumber": 18,
  "warnings": []
}
```


5. Produce some events into readings topic

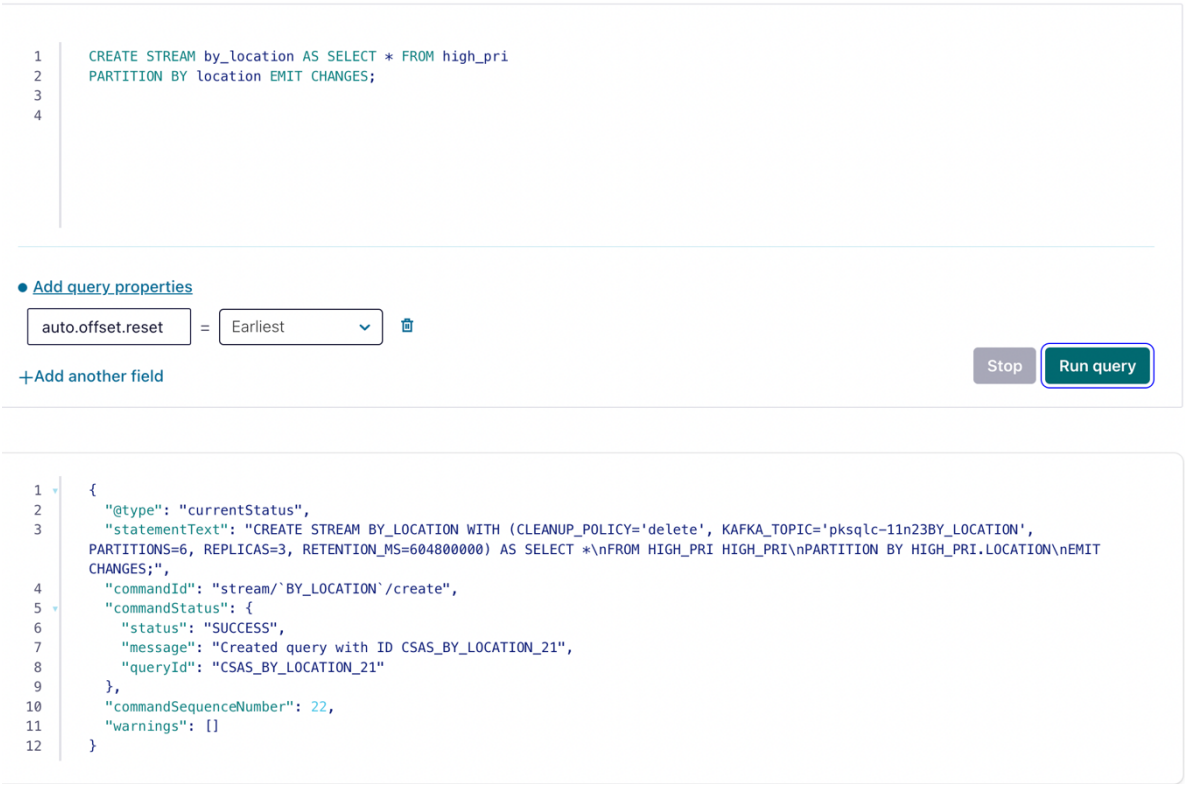
INSERT INTO readings (sensor, val, location) VALUES ('sensor-3', 36, 'motor');

6. Run some queries

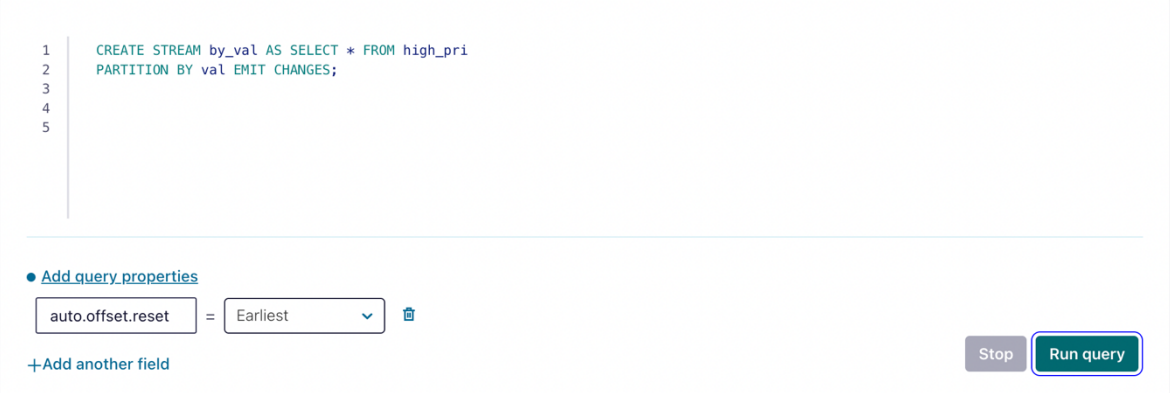
CREATE STREAM high_pri AS SELECT sensor, val, UCASE(location) as location
FROM readings where val > 42 EMIT CHANGES;

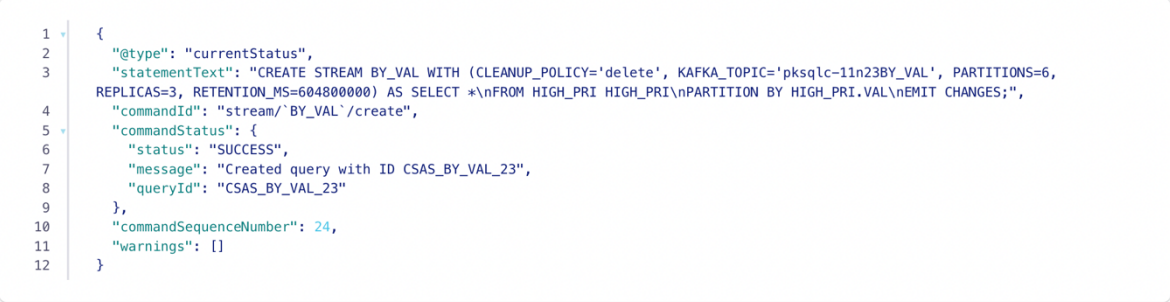


CREATE STREAM by_location AS SELECT * FROM high_pri
PARTITION BY location EMIT CHANGES;



CREATE STREAM by_val AS SELECT * FROM high_pri
PARTITION BY val EMIT CHANGES;

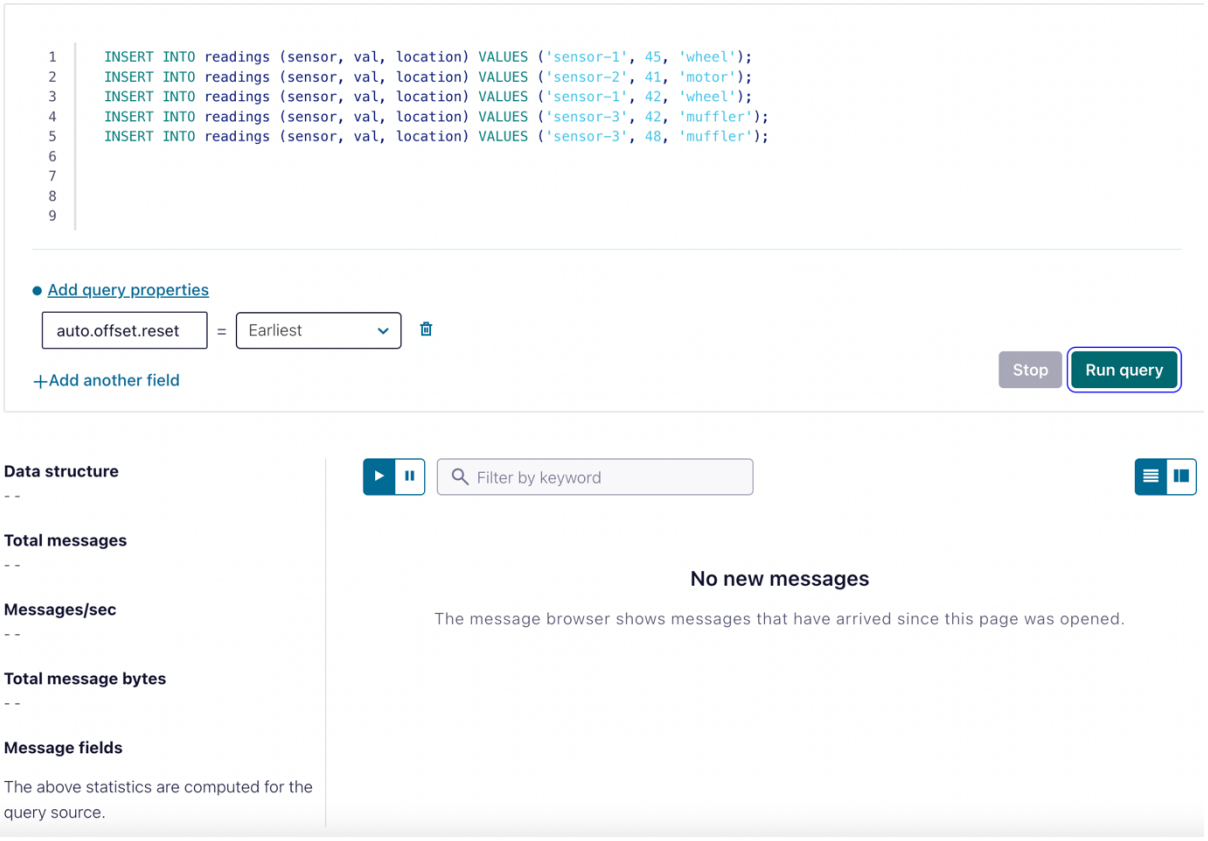




Lab 5

1. Considering the readings topic is already created
readings --> partitions 6

2. Produce some events into readings topic
INSERT INTO readings (sensor, val, location) VALUES ('sensor-1', 45, 'wheel');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-2', 41, 'motor');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-1', 42, 'wheel');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-3', 42, 'muffler');
INSERT INTO readings (sensor, val, location) VALUES ('sensor-3', 48, 'muffler');



3. Create KTable:

```
CREATE TABLE avg_readings AS
        SELECT sensor, AVG(val) as avg
        FROM readings
        GROUP BY sensor
        EMIT CHANGES;
```

```
CREATE TABLE avg_readings AS
    SELECT sensor, AVG(val) as avg
    FROM readings
    GROUP BY sensor
    EMIT CHANGES;
```

Add query properties

auto.offset.reset = Earliest

+Add another field

Stop | Run query


```
{
  "@type": "currentStatus",
  "statementText": "CREATE TABLE AVG_READINGS WITH (CLEANUP_POLICY='compact', KAFKA_TOPIC='pksqlc-11n23AVG_READINGS', PARTITIONS=6, REPLICAS=3, RETENTION_MS=604800000) AS SELECT\n  READINGS.SENSOR SENSOR,\n  AVG(READINGS.VAL) AVG\nFROM READINGS READINGS\nGROUP BY READINGS.SENSOR\nEMIT CHANGES;",
  "commandId": "table/`AVG_READINGS`/create",
  "commandStatus": {
    "status": "SUCCESS",
    "message": "Created query with ID CTAS_AVG_READINGS_25",
    "queryId": "CTAS_AVG_READINGS_25"
  },
  "commandSequenceNumber": 26,
  "warnings": []
}
```


CREATE TABLE part_avg AS
SELECT location, AVG(val) as avg
FROM readings
GROUP BY location
EMIT CHANGES;

```
CREATE TABLE part_avg AS
    SELECT location, AVG(val) as avg
    FROM readings
    GROUP BY location
    EMIT CHANGES;
```

Add query properties

auto.offset.reset = Earliest

+Add another field

Stop | Run query


```
{
  "@type": "currentStatus",
  "statementText": "CREATE TABLE PART_AVG WITH (CLEANUP_POLICY='compact', KAFKA_TOPIC='pksqlc-11n23PART_AVG', PARTITIONS=6, REPLICAS=3, RETENTION_MS=604800000) AS SELECT\n  READINGS.LOCATION LOCATION,\n  AVG(READINGS.VAL) AVG\nFROM READINGS READINGS\nGROUP BY READINGS.LOCATION\nEMIT CHANGES;",
  "commandId": "table/`PART_AVG`/create",
  "commandStatus": {
    "status": "SUCCESS",
    "message": "Created query with ID CTAS_PART_AVG_27",
    "queryId": "CTAS_PART_AVG_27"
  },
  "commandSequenceNumber": 28,
  "warnings": []
}
```