SAKARYA
ÜNİVERSİTESİ

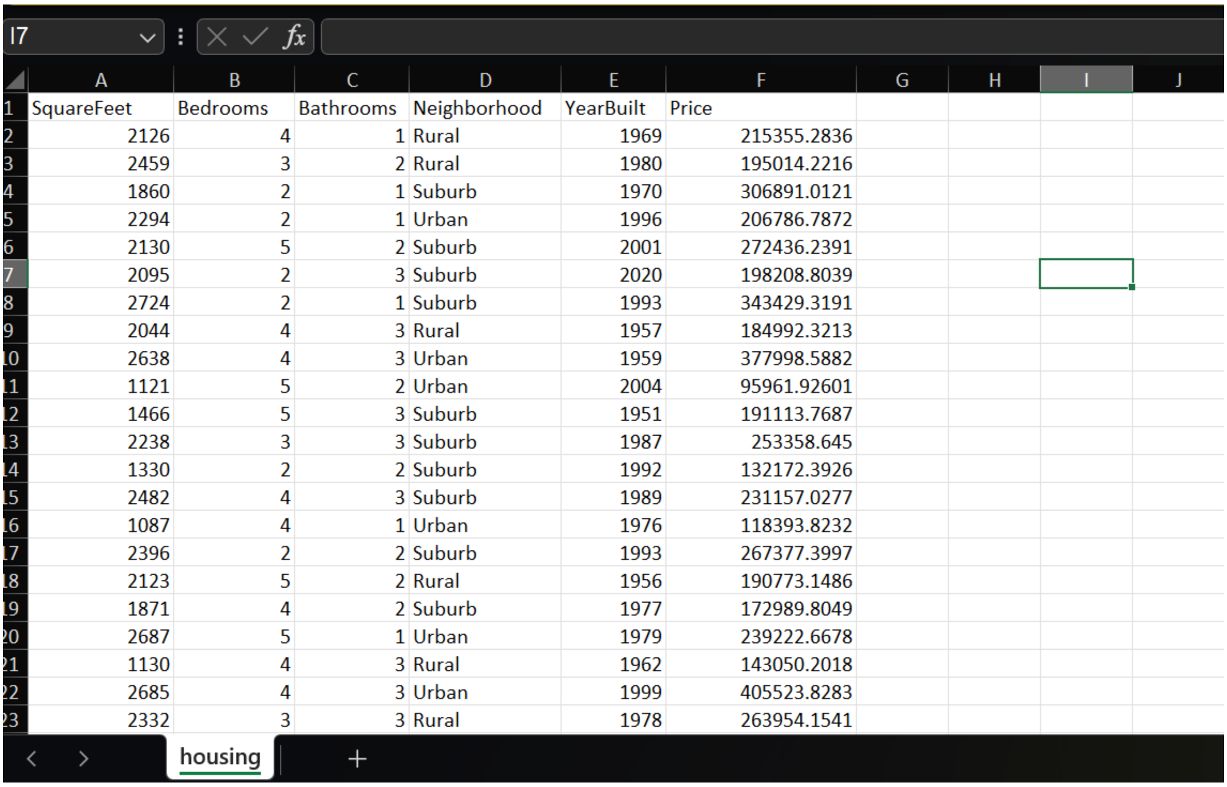
BİLGİSAYAR-BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLHİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Büyük Veriye Giriş dersi 2023-2024 Güz dönemi Proje raporu

Hazırladı: Fuad Garibli
Öğrenci Numarası: G201210558
Şube numarası: 2A

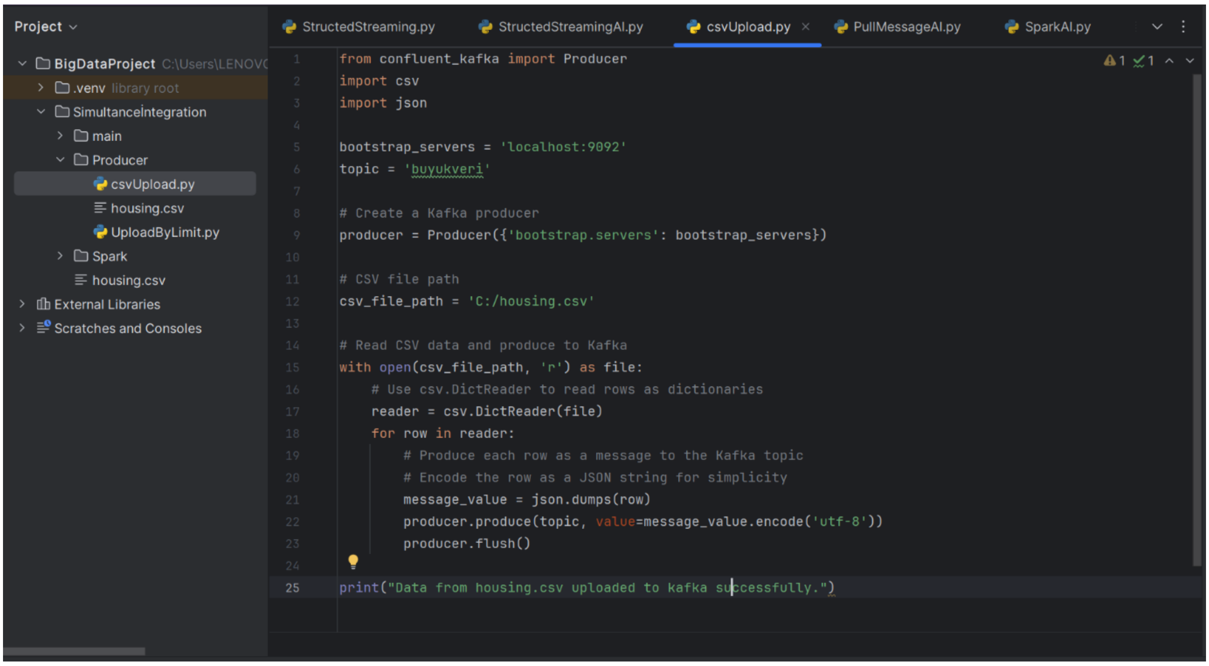
Bizden istenen Kafka ve Spark kullanılarak, gerçek zamanlı verilerin işlenmesine yönelik bir işlem hattını göstermektedir. Sistem, veri üretimi, kafka mesaj üretimi, Spark Yapılandırılmış Akış (structed streaming), Spark makine öğrenmesi (Spark ML) ve pipeline entegrasyonu için bileşenler içerir.

1) Öncelikle Veri setimizi tanıtalım:
   Veri setimiz totalde 1510 satırdan ve 6 sütundan oluşan bir ev fiyatları listesidir. Kanadada Vancouver eyaletinden toplanmış bir verisetidir (housing.csv). .CSV dosya formatında olup içinde comma seperated values (noktayla ayrılmış veriler) vardır:

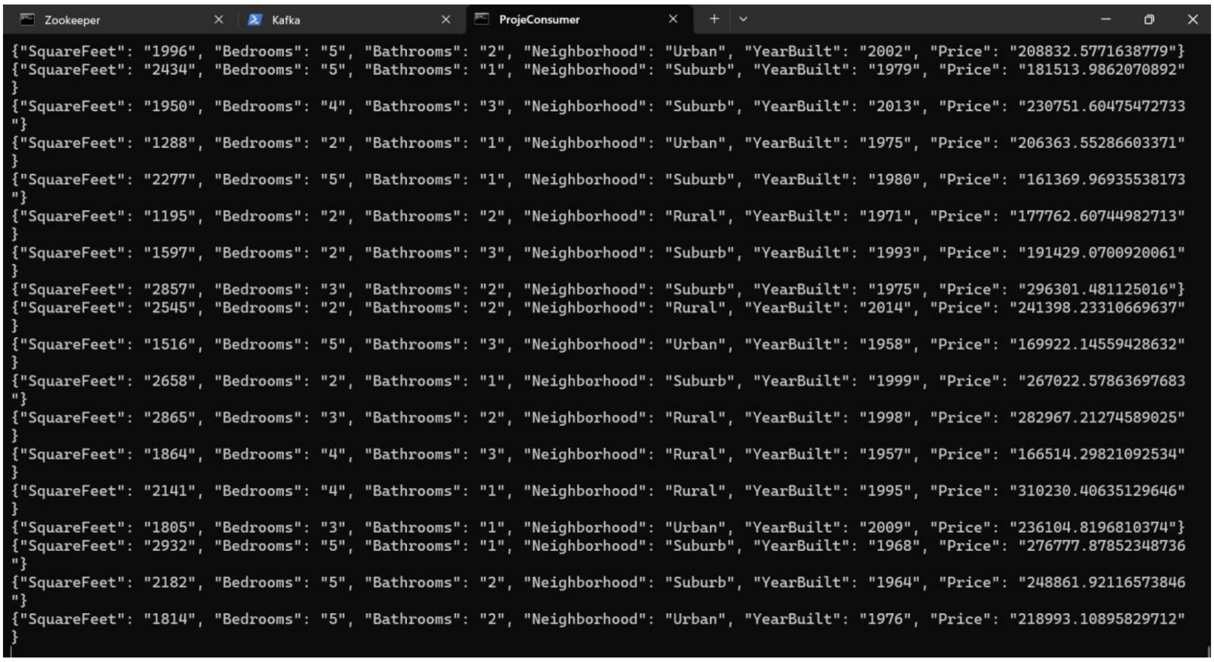

I7
fx
housing

	A	B	C	D	E	F
1	SquareFeet	Bedrooms	Bathrooms	Neighborhood	YearBuilt	Price
2	2126	4	1	Rural	1969	215355.2836
3	2459	3	2	Rural	1980	195014.2216
4	1860	2	1	Suburb	1970	306891.0121
5	2294	2	1	Urban	1996	206786.7872
6	2130	5	2	Suburb	2001	272436.2391
7	2095	2	3	Suburb	2020	198208.8039
8	2724	2	1	Suburb	1993	343429.3191
9	2044	4	3	Rural	1957	184992.3213
10	2638	4	3	Urban	1959	377998.5882
11	1121	5	2	Urban	2004	95961.92601
12	1466	5	3	Suburb	1951	191113.7687
13	2238	3	3	Suburb	1987	253358.645
14	1330	2	2	Suburb	1992	132172.3926
15	2482	4	3	Suburb	1989	231157.0277
16	1087	4	1	Urban	1976	118393.8232
17	2396	2	2	Suburb	1993	267377.3997
18	2123	5	2	Rural	1956	190773.1486
19	1871	4	2	Suburb	1977	172989.8049
20	2687	5	1	Urban	1979	239222.6678
21	1130	4	3	Rural	1962	143050.2018
22	2685	4	3	Urban	1999	405523.8283
23	2332	3	3	Rural	1978	263954.1541

2) Bu veri setini, PyCharmda oluşturduğumuz "csvUpload.py" dosyasında olan kodlar sayesinde kafka producere aktarıyoruz.

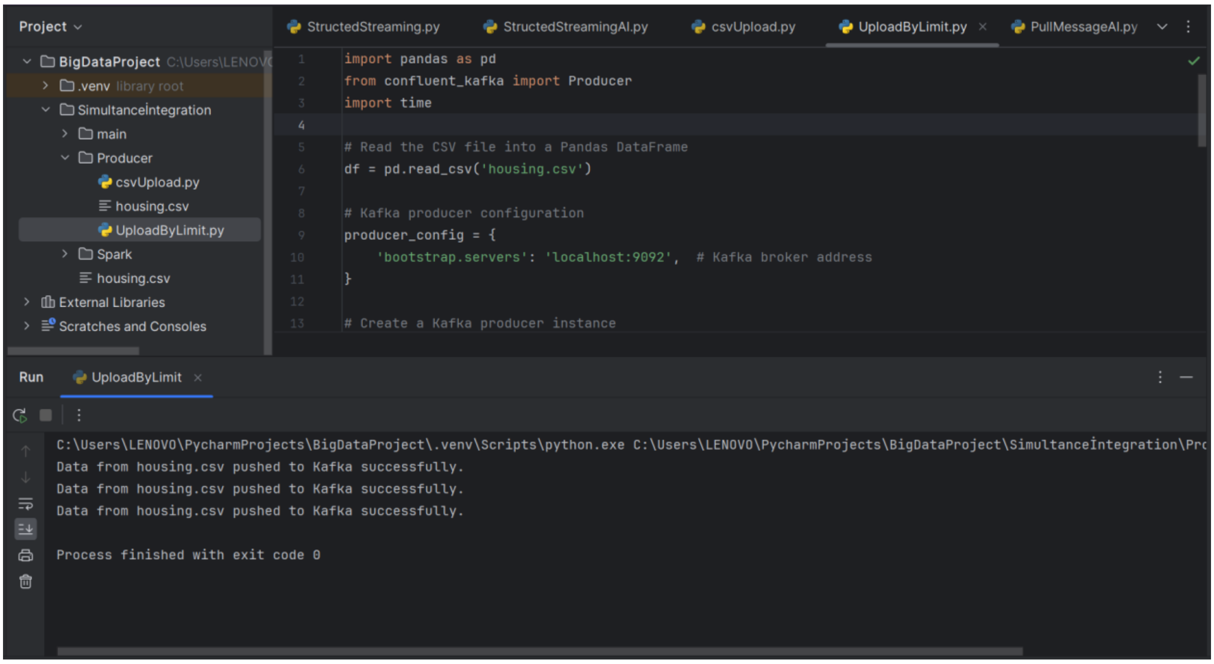


```python
from confluent_kafka import Producer
import csv
import json

bootstrap_servers = 'localhost:9092'
topic = 'buyukveri'

# Create a Kafka producer
producer = Producer({'bootstrap.servers': bootstrap_servers})

# CSV file path
csv_file_path = 'C:/housing.csv'

# Read CSV data and produce to Kafka
with open(csv_file_path, 'r') as file:
    # Use csv.DictReader to read rows as dictionaries
    reader = csv.DictReader(file)
    for row in reader:
        # Produce each row as a message to the Kafka topic
        # Encode the row as a JSON string for simplicity
        message_value = json.dumps(row)
        producer.produce(topic, value=message_value.encode('utf-8'))
        producer.flush()

print("Data from housing.csv uploaded to kafka successfully.")
```



```
{"SquareFeet": "1996", "Bedrooms": "5", "Bathrooms": "2", "Neighborhood": "Urban", "YearBuilt": "2002", "Price": "208832.5771638779"}
{"SquareFeet": "2434", "Bedrooms": "5", "Bathrooms": "1", "Neighborhood": "Suburb", "YearBuilt": "1979", "Price": "181513.9862070892"
}
{"SquareFeet": "1950", "Bedrooms": "4", "Bathrooms": "3", "Neighborhood": "Suburb", "YearBuilt": "2013", "Price": "230751.60475472733
"}
{"SquareFeet": "1288", "Bedrooms": "2", "Bathrooms": "1", "Neighborhood": "Urban", "YearBuilt": "1975", "Price": "206363.55286603371"
}
{"SquareFeet": "2277", "Bedrooms": "5", "Bathrooms": "1", "Neighborhood": "Suburb", "YearBuilt": "1980", "Price": "161369.96935538173
"}
{"SquareFeet": "1195", "Bedrooms": "2", "Bathrooms": "2", "Neighborhood": "Rural", "YearBuilt": "1971", "Price": "177762.60744982713"
}
{"SquareFeet": "1597", "Bedrooms": "2", "Bathrooms": "3", "Neighborhood": "Suburb", "YearBuilt": "1993", "Price": "191429.0700920061"
}
{"SquareFeet": "2857", "Bedrooms": "3", "Bathrooms": "2", "Neighborhood": "Suburb", "YearBuilt": "1975", "Price": "296301.481125016"}
{"SquareFeet": "2545", "Bedrooms": "2", "Bathrooms": "2", "Neighborhood": "Rural", "YearBuilt": "2014", "Price": "241398.23310669637"
}
{"SquareFeet": "1516", "Bedrooms": "5", "Bathrooms": "3", "Neighborhood": "Urban", "YearBuilt": "1958", "Price": "169922.14559428632"
}
{"SquareFeet": "2658", "Bedrooms": "2", "Bathrooms": "1", "Neighborhood": "Suburb", "YearBuilt": "1999", "Price": "267022.57863697683
"}
{"SquareFeet": "2865", "Bedrooms": "3", "Bathrooms": "2", "Neighborhood": "Rural", "YearBuilt": "1998", "Price": "282967.21274589025"
}
{"SquareFeet": "1864", "Bedrooms": "4", "Bathrooms": "3", "Neighborhood": "Rural", "YearBuilt": "1957", "Price": "166514.29821092534"
}
{"SquareFeet": "2141", "Bedrooms": "4", "Bathrooms": "1", "Neighborhood": "Rural", "YearBuilt": "1995", "Price": "310230.40635129646"
}
{"SquareFeet": "1805", "Bedrooms": "3", "Bathrooms": "1", "Neighborhood": "Urban", "YearBuilt": "2009", "Price": "236104.8196810374"}
{"SquareFeet": "2932", "Bedrooms": "5", "Bathrooms": "1", "Neighborhood": "Suburb", "YearBuilt": "1968", "Price": "276777.87852348736
"}
{"SquareFeet": "2182", "Bedrooms": "5", "Bathrooms": "2", "Neighborhood": "Suburb", "YearBuilt": "1964", "Price": "248861.92116573846
"}
{"SquareFeet": "1814", "Bedrooms": "5", "Bathrooms": "2", "Neighborhood": "Urban", "YearBuilt": "1976", "Price": "218993.10895829712"
}
```

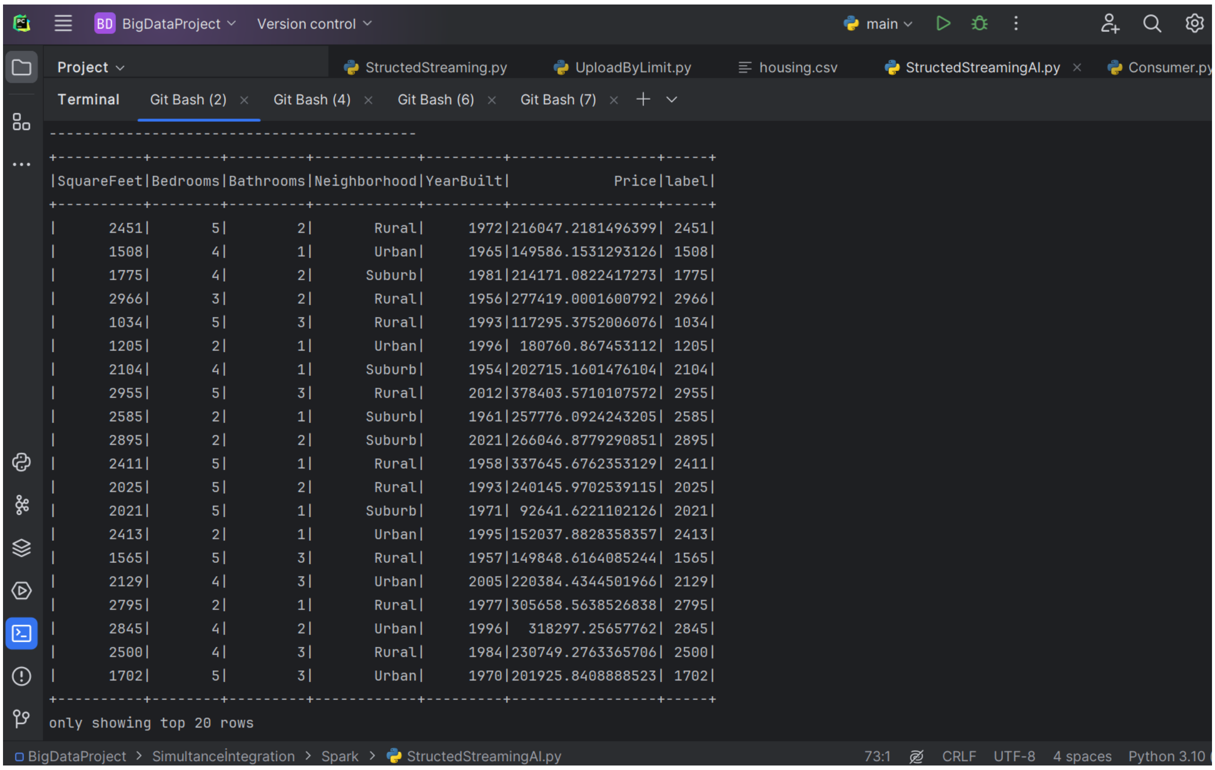
3)Veri üretiminin süresini kontrol etmek için UploadByLimit.py scripti, belirli bir süre boyunca Kafka'ya veri gönderir. Bu, Kafka'ya aktarılan veri miktarını test etmek ve sınırlamak için kullanılacaktır. Bu süre zarfında Streaming yapabilecek duruma geliyoruz:



```
import pandas as pd
from confluent_kafka import Producer
import time

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv('housing.csv')

# Kafka producer configuration
producer_config = {
    'bootstrap.servers': 'localhost:9092',  # Kafka broker address
}

# Create a Kafka producer instance
```

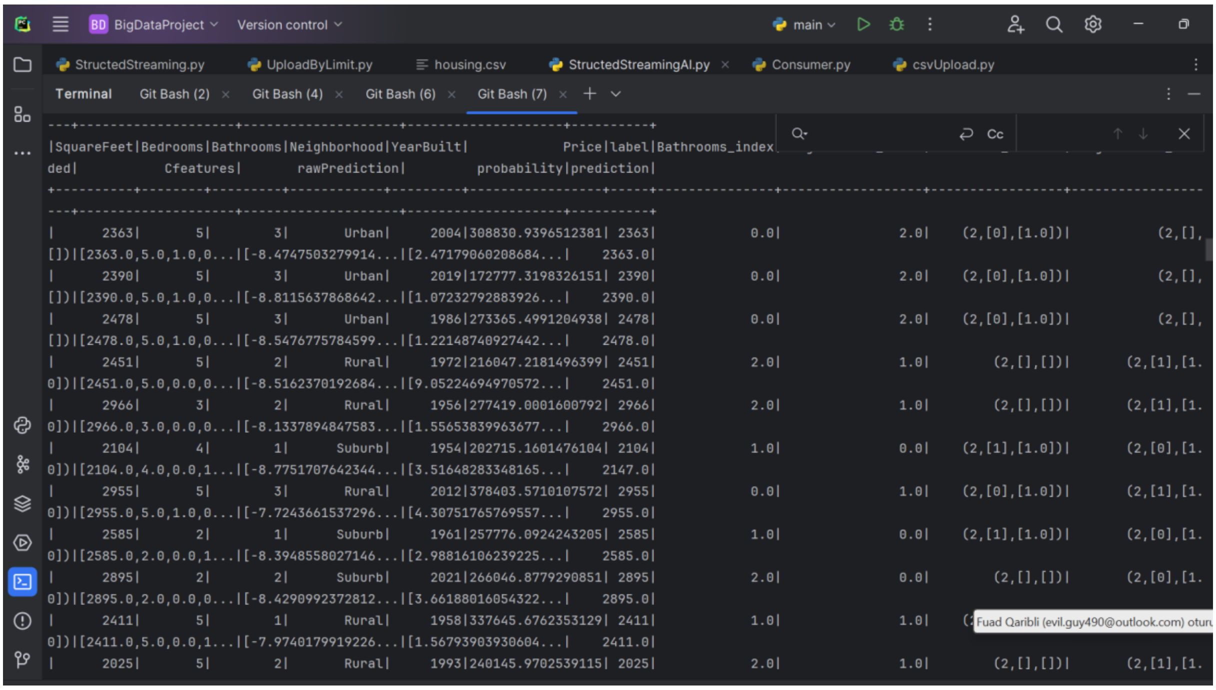
```
C:\Users\LENOVO\PycharmProjects\BigDataProject\.venv\Scripts\python.exe C:\Users\LENOVO\PycharmProjects\BigDataProject\Simultanceİntegration\Pro
Data from housing.csv pushed to Kafka successfully.
Data from housing.csv pushed to Kafka successfully.
Data from housing.csv pushed to Kafka successfully.

Process finished with exit code 0
```

3) StructedStreaming.py çalıştığı zaman artık veriler kafka topic'e yüklenerek bizim Streamingimiz gerçekleştiriyor. Buna uygun çıktı aşağıda gösterilmiştir:



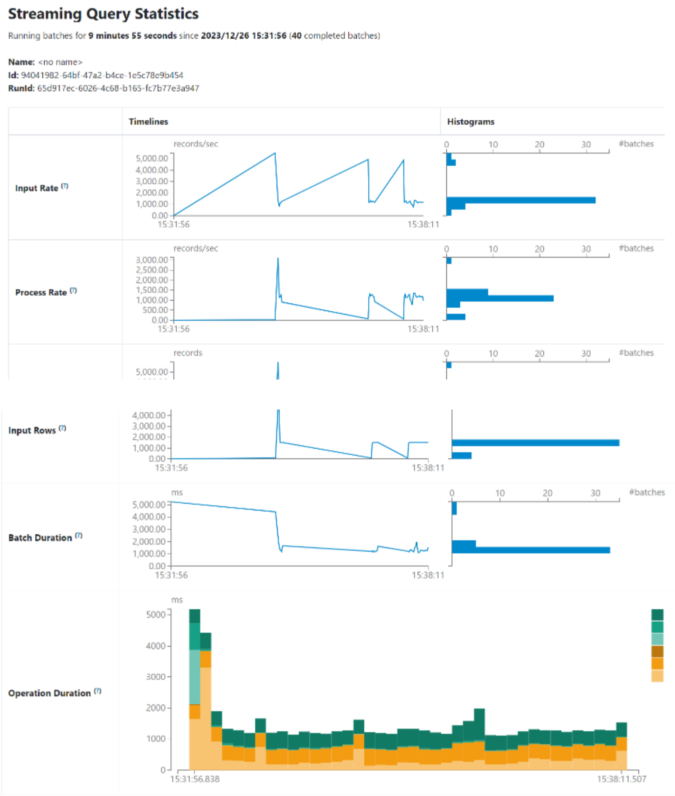
```
-------------------------------------------
+----------+--------+---------+------------+---------+-----------------+-----+
|SquareFeet|Bedrooms|Bathrooms|Neighborhood|YearBuilt|            Price|label|
+----------+--------+---------+------------+---------+-----------------+-----+
|      2451|       5|        2|       Rural|     1972|216047.2181496399| 2451|
|      1508|       4|        1|       Urban|     1965|149586.1531293126| 1508|
|      1775|       4|        2|      Suburb|     1981|214171.0822417273| 1775|
|      2966|       3|        2|       Rural|     1956|277419.0001600792| 2966|
|      1034|       5|        3|       Rural|     1993|117295.3752006076| 1034|
|      1205|       2|        1|       Urban|     1996| 180760.867453112| 1205|
|      2104|       4|        1|      Suburb|     1954|202715.1601476104| 2104|
|      2955|       5|        3|       Rural|     2012|378403.5710107572| 2955|
|      2585|       2|        1|      Suburb|     1961|257776.0924243205| 2585|
|      2895|       2|        2|      Suburb|     2021|266046.8779290851| 2895|
|      2411|       5|        1|       Rural|     1958|337645.6762353129| 2411|
|      2025|       5|        2|       Rural|     1993|240145.9702539115| 2025|
|      2021|       5|        1|      Suburb|     1971| 92641.6221102126| 2021|
|      2413|       2|        1|       Urban|     1995|152037.8828358357| 2413|
|      1565|       5|        3|       Rural|     1957|149848.6164085244| 1565|
|      2129|       4|        3|       Urban|     2005|220384.4344501966| 2129|
|      2795|       2|        1|       Rural|     1977|305658.5638526838| 2795|
|      2845|       4|        2|       Urban|     1996|  318297.25657762| 2845|
|      2500|       4|        3|       Rural|     1984|230749.2763365706| 2500|
|      1702|       5|        3|       Urban|     1970|201925.8408888523| 1702|
+----------+--------+---------+------------+---------+-----------------+-----+
only showing top 20 rows
```

4) StructedStreamingAI.py dosyası yukarda tanımlanan işlemleri yaparak, üstüne Machine Learning tekniklerini kullanarak bize herhangi bir senede inşa edilmiş evin alanına göre fiyatını tahmin ediyor. Çıktısı aşağıda verilmiştir:



```
---+-------------------+-------------------+-------------------+---------+
|SquareFeet|Bedrooms|Bathrooms|Neighborhood|YearBuilt|           Price|label|Bathrooms_index
ded|          Cfeatures|       rawPrediction|         probability|prediction|
+---------+--------+---------+------------+---------+-----------------+-----+---------------+-----------------+-------------------+------------------
---+-------------------+-------------------+-------------------+---------+
|      2363|       5|        3|       Urban|     2004|308830.9396512381| 2363|            0.0|                2.0|     (2,[0],[1.0])|         (2,[],
[])|[2363.0,5.0,1.0,0...|[-8.4747503279914...|[2.47179060208684...|     2363.0|
|      2390|       5|        3|       Urban|     2019|172777.3198326151| 2390|            0.0|                2.0|     (2,[0],[1.0])|         (2,[],
[])|[2390.0,5.0,1.0,0...|[-8.8115637868642...|[1.07232792883926...|     2390.0|
|      2478|       5|        3|       Urban|     1986|273365.4991204938| 2478|            0.0|                2.0|     (2,[0],[1.0])|         (2,[],
[])|[2478.0,5.0,1.0,0...|[-8.5476775784599...|[1.22148740927442...|     2478.0|
|      2451|       5|        2|       Rural|     1972|216047.2181496399| 2451|            2.0|                1.0|         (2,[],[])|     (2,[1],[1.
0])|[2451.0,5.0,0.0,0...|[-8.5162370192684...|[9.05224694970572...|     2451.0|
|      2966|       3|        2|       Rural|     1956|277419.0001600792| 2966|            2.0|                1.0|         (2,[],[])|     (2,[1],[1.
0])|[2966.0,3.0,0.0,0...|[-8.1337894847583...|[1.55653839963677...|     2966.0|
|      2104|       4|        1|      Suburb|     1954|202715.1601476104| 2104|            1.0|                0.0|     (2,[1],[1.0])|     (2,[0],[1.
0])|[2104.0,4.0,0.0,1...|[-8.7751707642344...|[3.51648283348165...|     2147.0|
|      2955|       5|        3|       Rural|     2012|378403.5710107572| 2955|            0.0|                1.0|     (2,[0],[1.0])|     (2,[1],[1.
0])|[2955.0,5.0,1.0,0...|[-7.7243661537296...|[4.30751765769557...|     2955.0|
|      2585|       2|        1|      Suburb|     1961|257776.0924243205| 2585|            1.0|                0.0|     (2,[1],[1.0])|     (2,[0],[1.
0])|[2585.0,2.0,0.0,1...|[-8.3948558027146...|[2.98816106239225...|     2585.0|
|      2895|       2|        2|      Suburb|     2021|266046.8779290851| 2895|            2.0|                0.0|         (2,[],[])|     (2,[0],[1.
0])|[2895.0,2.0,0.0,0...|[-8.4290992372812...|[3.66188016054322...|     2895.0|
|      2411|       5|        1|       Rural|     1958|337645.6762353129| 2411|            1.0|                1.0|     (2
0])|[2411.0,5.0,0.0,1...|[-7.9740179919226...|[1.56793903930604...|     2411.0|
|      2025|       5|        2|       Rural|     1993|240145.9702539115| 2025|            2.0|                1.0|         (2,[],[])|     (2,[1],[1.
```

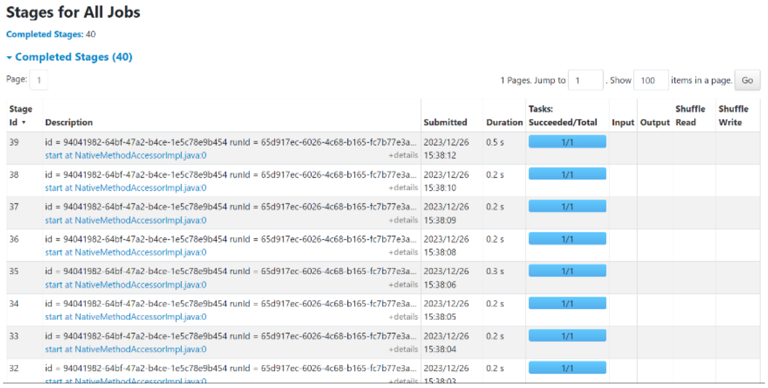
5) Bazı Spark LocalHost çıktıları aşağıda verilmiştir:



Streaming Query Statistics

Running batches for **9 minutes 55 seconds** since **2023/12/26 15:31:56** (**40** completed batches)

Name: <no name>
Id: 94041982-64bf-47a2-b4ce-1e5c78e9b454
RunId: 65d917ec-6026-4c68-b165-fc7b77e3a947



Stages for All Jobs

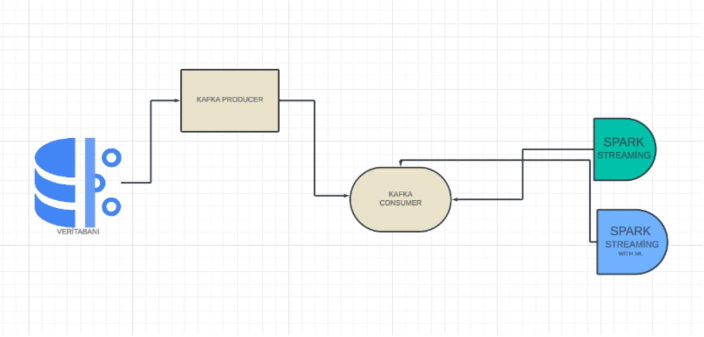
Completed Stages: 40

▾ **Completed Stages (40)**

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
39	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:12	0.5 s	1/1				
38	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:10	0.2 s	1/1				
37	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:09	0.2 s	1/1				
36	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:08	0.2 s	1/1				
35	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:06	0.3 s	1/1				
34	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:05	0.2 s	1/1				
33	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:04	0.2 s	1/1				
32	id = 94041982-64bf-47a2-b4ce-1e5c78e9b454 runId = 65d917ec-6026-4c68-b165-fc7b77e3a... start at NativeMethodAccessorImpl.java:0 +details	2023/12/26 15:38:03	0.2 s	1/1				

6) Akış diagramı:


KAFKA PRODUCER
VERİTABANI
KAFKA CONSUMER
SPARK STREAMİNG
SPARK STREAMİNG WITH ML