# MongoDB Assignment

## 1. Kafka Producer in Python

- Develop a Python script to act as a Kafka producer.
- Use Pandas to read logistics data from a CSV file.
- Serialize the data into Avro format and publish it to a Confluent Kafka topic.

```python
        if dict_data[key] in ['nan', 0, 'null', 'NULL']:
            dict_data[key] = None
        elif key in ['BookingID_Date', 'actual_eta', 'trip_start_date', 'trip_end_date']:
            datetime_obj = datetime.strptime(dict_data[key], __format: '%Y-%m-%d %H:%M:%S')
            dict_data[key] = datetime_obj.strftime("%d-%m-%Y, %H:%M:%S")
    print(dict_data)
    return dict_data
```

```
{'GpsProvider': 'VIRSTEMPO', 'BookingID': 'VCV00010665/082021', 'Market/Regular ': 'Regular', 'BookingID_Date': '21-08-2020, 00:00:00', 'vehicle_no': 'HR55AA5092', 'Origin_Location': 'HEROMOTOC
User record b'725' successfully produced to delivery_trip_truck_data [1] at offset 134
{'GpsProvider': 'CONSENT TRACK', 'BookingID': 'MVCV0001094/082021', 'Market/Regular ': 'Market', 'BookingID_Date': '20-08-2020, 00:00:00', 'vehicle_no': 'KA27A4080', 'Origin_Location': 'ASHOK L
User record b'726' successfully produced to delivery_trip_truck_data [1] at offset 135
{'GpsProvider': 'CONSENT TRACK', 'BookingID': 'VCV00010014/082021', 'Market/Regular ': 'Regular', 'BookingID_Date': '20-08-2020, 00:00:00', 'vehicle_no': 'TN23AA5466', 'Origin_Location': 'ASHOK
User record b'727' successfully produced to delivery_trip_truck_data [5] at offset 126
{'GpsProvider': 'VAMOSYS', 'BookingID': 'VCV00010572/082021', 'Market/Regular ': 'Regular', 'BookingID_Date': '21-08-2020, 00:00:00', 'vehicle_no': 'TN18m5881', 'Origin_Location': 'TRACTORS & F
User record b'728' successfully produced to delivery_trip_truck_data [0] at offset 128
{'GpsProvider': 'VAMOSYS', 'BookingID': 'VCV00009694/082021', 'Market/Regular ': 'Regular', 'BookingID_Date': '20-08-2020, 00:00:00', 'vehicle_no': 'TN18m5881', 'Origin_Location': 'TRACTORS & F
User record b'729' successfully produced to delivery_trip_truck_data [4] at offset 118
```

# delivery_trip_truck_data

Actions

Overview | Messages | Schema | Configuration

| Production in last hour | Consumption in last hour | Total messages | Retention time |
|---|---|---|---|
| 643 messages | -- messages | 792 | 1 week |

Filter by timestamp, offset, key or value | All partitions | Latest | Max 50 results

50 messages shown

Auto-refresh on

CSV JSON

| Timestamp | Offset | Partition | Key | Value |
|---|---|---|---|---|
| 1721361179365 | 141 | 0 | 794 | {"GpsProvider":{"string":"VAMOSYS"},"BookingID":{"string":"VCV00010187/082021"},"Market_Regular":null,"BookingID_Date":{"s... |
| 1721361178658 | 148 | 1 | 793 | {"GpsProvider":{"string":"CONSENT TRACK"},"BookingID":{"string":"VCV00008504/082021"},"Market_Regular":null,"BookingID... |
| 1721361178091 | 147 | 1 | 792 | {"GpsProvider":{"string":"BEECON"},"BookingID":{"string":"VCV00008832/082021"},"Market_Regular":null,"BookingID_Date":{"st... |
| 1721361177542 | 137 | 5 | 791 | {"GpsProvider":{"string":"VAMOSYS"},"BookingID":{"string":"VCV00010203/082021"},"Market_Regular":null,"BookingID_Date":{"... |
| 1721361177003 | 136 | 5 | 790 | {"GpsProvider":{"string":"VAMOSYS"},"BookingID":{"string":"VCV00010216/082021"},"Market_Regular":null,"BookingID_Date":{"... |

## 2. Schema Registry Integration

- Establish a Schema Registry for managing Avro schemas.
- Ensure that the Kafka producer and consumer fetch the schema from the Schema Registry during serialization and deserialization.

## 4. Kafka Consumer in Python

- Write a Python script for the Kafka consumer.
- Deserialize the Avro data and ingest it into a MongoDB collection.

## default

Clusters | Flink New | Network management | Schema registry

Search by schema subject name | Compatibility mode: BACKWARD | + Add schema

| Subject | Schema type | Current version |
|---|---|---|
| delivery_trip_truck_data-key | Avro | 1 |
| delivery_trip_truck_data-value | Avro | 1 |

Projec. | kafka_consumer_csv_data.py | config.properties | main.py | kafka_producer_csv_data.py | python_flask_rest_api.py | rest_api_for_mongodb_wi

C:\SparkPython\pyspark
C:\SparkPython\python_
C:\Users\DELL\Pycharm
D:\Grow_Data_Skills\kaf
.idea
kafka_miongodb_assign

```python
consumer.subscribe(topics=[config['TOPIC_CONF']['topic.name']], on_assign=assign_callback)
consumer_id = random.randint( a: 125058,  b: 125589)
try:
    while True:
        msg = consumer.poll(1.0)  # How many seconds to wait for message
```

assign_callback()

Run | kafka_consumer_csv_data | kafka_producer_csv_data


```
"Data_Ping_time": "00:25:07",
"Planned_ETA": "00:26:31",
"Current_Location": "NH 48, Elathagiri, Tamil Nadu 635108, India",
"DestinationLocation": "ASHOK LEYLAND PARTS-HOSUR,HOSUR,TAMIL NADU",
"actual_eta": "2020-08-19 12:15:00",
"Curr_lat": 12.534722,
"Curr_lon": 78.299446,
"ontime": null,
"delay": "R",
"OriginLocation_Code": "CHEENNALLCCA1",
"DestinationLocation_Code": "HOSCHAALLCCA2",
"trip_start_date": "2020-08-13 10:26:00",
"trip_end_date": "2020-08-13 10:27:00",
"TRANSPORTATION_DISTANCE_IN_KM": 350.0,
"vehicleType": null,
"Minimum_kms_to_be_covered_in_a_day": null,
"Driver_Name": null,
```

# delivery_trip_truck_data

Query with Flink

Share

Overview Messages Schema Configuration

## Production

1.07K

Bytes per second

## Consumption

2.44K

Bytes per second

mongoDbLocal > kafka_delivery_trip > truck_data

Documents 38 | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

Explain | Reset | Find | Options

ADD DATA | EXPORT DATA | UPDATE | DELETE

1 - 20 of 38


```
_id: ObjectId('6699e349490ff0a42faec5a4')
GpsProvider : "CONSENT TRACK"
BookingID : "AEIBK2026772"
Market_Regular : null
BookingID_Date : 2020-08-20T00:00:00.000+00:00
vehicle_no : "UP17T8250"
Origin_Location : "Ramte Ram Road, Ghaziabad, Uttar Pradesh"
Destination_Location : "Muzakkipur, Meerut, Uttar Pradesh"
Org_lat_lon : "28.658633,77.438643"
Des_lat_lon : "28.91386,77.54849"
Data_Ping_time : "00:55:09"
Planned_ETA : "00:44:33"
Current_Location : "17, Mainapur, Ghaziabad, Uttar Pradesh 201003, India"
DestinationLocation : "Muzakkipur, Meerut, Uttar Pradesh"
actual_eta : 2020-08-25T17:31:00.000+00:00
Curr_lat : 28.711
Curr_lon : 77.4591
ontime : null
delay : "R"
OriginLocation_Code : "V0045769 "
DestinationLocation_Code : "LE005786"
trip_start_date : 2020-08-25T13:14:00.000+00:00
trip_end_date : 2020-08-25T16:42:00.000+00:00
TRANSPORTATION_DISTANCE_IN_KM : 35
vehicleType : "40 FT 3XL Trailer 35MT"
```


SHOW 8 MORE FIELDS

6. **Data Validation in Kafka Consumer**

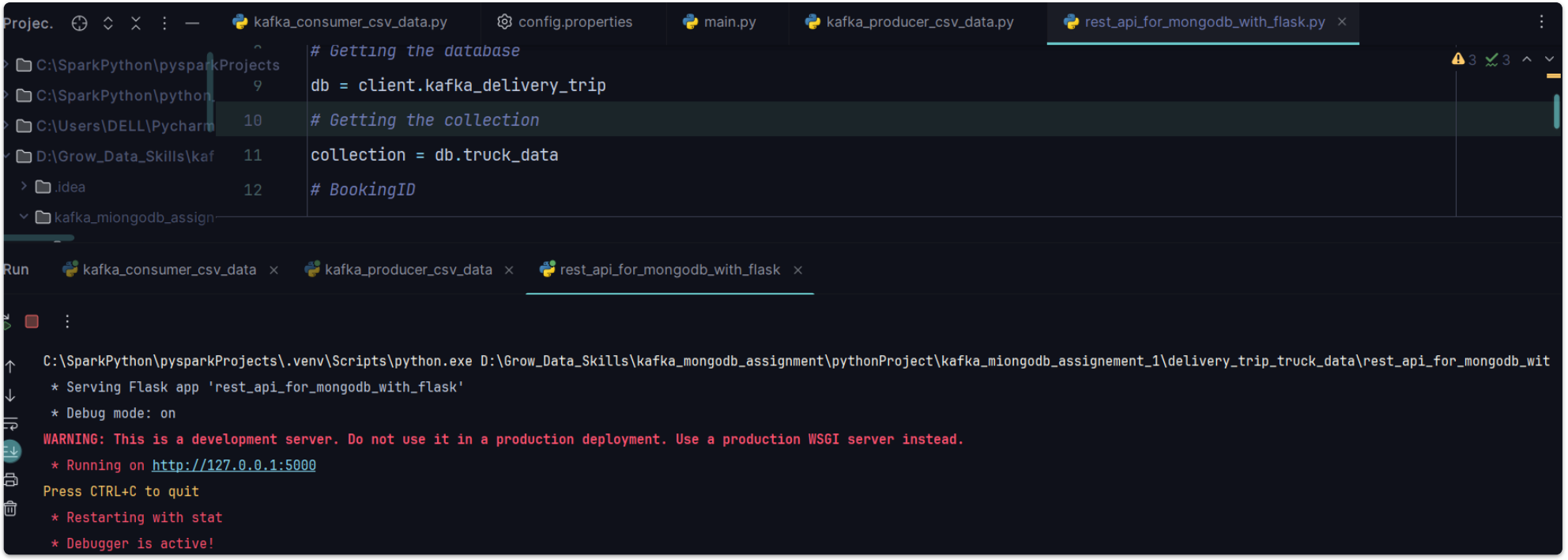
- Implement data validation checks in the consumer script before ingesting data into MongoDB.
- Validations like checking for null values, data type validation, and format checks.
- More assumptions can be taken for data validation, make sure to list down your assumptions in the submission document.

```python
def validate_data(data_dict):
    required_fields = [...]
    # Validate required fields
    for key in data_dict.keys():
        if isinstance(data_dict[key], dict):...
        if key not in required_fields and data_dict[key] is None:
            raise ValueError(f"{key} is missing in the data or value of that key is {None}")

    # Validate data type of the fields
    # Validate datatime data type of the fields
    date_fields = ['BookingID_Date', 'actual_eta', 'trip_start_date', 'trip_end_date']
    for key in date_fields:...
    # Validate int and float data type of the fields
    int_float_fields = ['TRANSPORTATION_DISTANCE_IN_KM', 'Minimum_kms_to_be_covered_in_a_day', 'Curr_lat', 'Curr_lon',
                        'Driver_MobileNo']
    for key in int_float_fields:...
    return data_dict
```

## 7. API Development using MongoDB Atlas

- Create an API to interact with the MongoDB collection.
- Implement endpoints for filtering specific JSON documents and for aggregating data.
- More assumptions & use-cases can be considered for API creation, make sure to list down your assumptions & use-cases in the submission document.




Get The Details of the records based on the BookingID

GET get_all_items | POST create_item | PUT update_item | DEL delete_item | GET get_all_items | UAT

travel_data_collection / **get_all_items**

Save | Share

GET http://127.0.0.1:5000/trip_details/AEIBK2026772 **Send**

Params | Authorization | Headers (6) | Body | Scripts | Tests | Settings | Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|---|---|---|---|
| Key | Value | Description | |

Body | Cookies | Headers (5) | Test Results

Status: 200 OK Time: 13 ms Size: 1.42 KB Save as example

Pretty | Raw | Preview | Visualize | JSON


```
{
    "BookingID": "AEIBK2026772",
    "BookingID_Date": "Thu, 20 Aug 2020 00:00:00 GMT",
    "Curr_lat": 28.711,
    "Curr_lon": 77.4591,
    "Current_Location": "17, Mainapur, Ghaziabad, Uttar Pradesh 201003, India",
    "Data_Ping_time": "00:55:09",
    "Des_lat_lon": "28.91386,77.54849",
    "DestinationLocation": "Muzakkipur, Meerut, Uttar Pradesh",
    "DestinationLocation_Code": "LE005786",
    "Destination_Location": "Muzakkipur, Meerut, Uttar Pradesh",
    "Driver_MobileNo": 9355651335,
    "Driver_Name": "Satendra Kumar Tyagi",
    "GpsProvider": "CONSENT TRACK",
    "Market_Regular": null,
    "Material_Shipped": null,
    "Minimum_kms_to_be_covered_in_a_day": 250.0,
    "Org_lat_lon": "28.658633,77.438643",
    "OriginLocation_Code": "V0045769 ",
    "Origin_Location": "Ramte Ram Road, Ghaziabad, Uttar Pradesh",
    "Planned_ETA": "00:44:33",
    "TRANSPORTATION_DISTANCE_IN_KM": 35.0,
    "actual_eta": "Tue, 25 Aug 2020 17:31:00 GMT",
    "customerID": "LTLEXMUM40",
    "customerNameCode": "Larsen & toubro limited"
```


Postbot | Runner | Start Proxy | Cookies | Vault | Trash

Get aggregation details for all GpsProviders

travel_data_collection / **Get_aggr_for_all_gpsProviders**

Save | Share

GET http://127.0.0.1:5000/GpsProvider Send

Params | Authorization | Headers (6) | Body | Scripts | Tests | Settings | Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|---|---|---|---|

Body | Cookies | Headers (5) | Test Results

Status: 200 OK Time: 15 ms Size: 616 B Save as example

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "_id": "CONSENT TRACK",
        "avg_kms": 149.98685376661743,
        "count": 677,
        "max_kms": 990.0,
        "min_kms": 12.0,
        "total_kms": 101541.1
    },
    {
        "_id": "FLEETX",
        "avg_kms": 31.96666666666667,
        "count": 6,
        "max_kms": 53.9,
        "min_kms": 13.1,
        "total_kms": 191.8
    },
    {
        "_id": "JTECH",
        "avg_kms": 16.116666666666667,
        "count": 6,
        "max_kms": 29.0,
        "min_kms": 7.6,
        "total_kms": 96.7
    }
]
```


Get aggregation details for givenGpsProviders

travel_data_collection / **Get_aggr_for_specific_gpsProviders**

Save | Share

**POST** http://127.0.0.1:5000/GpsProvider **Send**

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

```
{
    "GpsProvider" : "CONSENT TRACK"
}
```

Body | Cookies | Headers (5) | Test Results | Status: 200 OK | Time: 10 ms | Size: 308 B | Save as example

Pretty | Raw | Preview | Visualize | JSON

```
{
    "_id": "CONSENT TRACK",
    "avg_kms": 149.98685376661743,
    "count": 677,
    "max_kms": 990.0,
    "min_kms": 12.0,
    "total_kms": 101541.1
}
```