Here are two README files for the programs you provided. These files explain the functionality, setup, and usage of each program in a way that can be easily understood by others visiting your GitHub repository.

README for `RestKafkaProducerAviationEdgeAVRO16.py`

`RestKafkaProducerAviationEdgeAVRO16.py`

Overview

This program is a Kafka producer that retrieves flight data from the Aviation Edge API, formats it according to an Avro schema, and sends it to a specified Kafka topic. It is designed to handle real-time data on flights, making it suitable for applications requiring live flight information.

Key Features

- **Data Retrieval**: Pulls live flight data from the Aviation Edge API.
- **Data Transformation**: Organizes the data to match a specified Avro schema for consistent data structuring.
- **Kafka Integration**: Publishes flight data to a Kafka topic, enabling downstream applications to consume and process the data in real time.

Prerequisites

- **Kafka**: Ensure Kafka and the Kafka Schema Registry are running and accessible.
- **Python Packages**: Install the following packages:

  ```
  pip install requests confluent-kafka avro-python3
  ```

Program Structure

1. **Avro Schema Definition**: Defines the schema for flight data, including attributes like altitude, speed, direction, and various aircraft and flight identifiers.
2. **AvroProducer Setup**: Configures the Kafka producer with Avro serialization and connects to the Kafka server.
3. **Data Fetching**: Calls the Aviation Edge API to retrieve flight data.
4. **Data Publishing**: Publishes each flight record to a Kafka topic after formatting it according to the Avro schema.

Usage

1. Ensure Kafka is running and reachable on the specified IP and port.

2. Run the program:

```
python RestKafkaProducerAviationEdgeAVRO16.py
```

3. The program will periodically fetch flight data, process it, and publish it to the Kafka topic `AviationEdgeFlightTracker`.

Example Output

Each message sent to Kafka contains details such as:

- Flight coordinates, altitude, and direction
- Departure and arrival information
- Aircraft identifiers and airline codes
- Flight speed and status

Notes

- The program is set to make up to 10 API calls, with a 30-second interval between calls.
- Ensure that the Aviation Edge API key is valid and has sufficient quota.

README for `FlightTrackerV3-checkpoint.ipynb`

`FlightTrackerV3-checkpoint.ipynb`

Overview

This Jupyter Notebook connects to a Vertica database to query, process, and visualize live flight data. It filters flights based on certain conditions, transforms unit values, and displays the data on a map using Plotly and Mapbox, making it a powerful tool for tracking specific flights in real time.

Key Features

- **Database Querying**: Connects to a Vertica database to fetch detailed flight data.
- **Data Filtering and Transformation**: Filters data to show only flights with a specific airline and status, and converts units for altitude and speed.
- **Interactive Visualization**: Displays flight locations on a Plotly map with altitude-based coloring and detailed hover information.

Prerequisites

- **Vertica Database**: Ensure that the Vertica database is accessible with the required credentials.

- **Python Packages**: Install the following packages:

```
pip install verticapy plotly
```

- **Mapbox Access Token**: Obtain a Mapbox access token to use Mapbox tiles in the Plotly map.

Program Structure

1. **Database Connection**: Establishes a connection to the Vertica database using `verticapy`.
2. **Data Filtering**: Filters flight data for active flights (`system_status == 'en-route'`) operated by a specific airline (e.g., American Airlines with code `AAL`).
3. **Data Transformation**: Converts altitude from meters to feet and speed from km/h to knots for easier interpretation.
4. **Map Visualization**: Uses Plotly to visualize flights on a map, with hover details that include altitude, speed, and location.

Usage

1. Open the notebook in Jupyter Notebook or JupyterLab.
2. Execute each cell in sequence:
    - Connect to the Vertica database.
    - Query, filter, and transform the flight data.
    - Display the interactive map.
3. Explore the map to view live positions and details for the selected flights.

Example Output

The notebook outputs an interactive map showing flight data, with markers representing flight positions. Each marker displays:

- Airline code, geographic coordinates, and altitude in feet
- Horizontal and vertical speeds in knots
- Direction and flight status

Notes

- Replace the Mapbox access token with a valid token for visualization.
- Modify filter conditions (e.g., airline code) to track different sets of flights as needed.
