Take Home Test - Data Engineer

Please follow the guidelines below:

- You need to complete this test in a **maximum of 7 days** from the day you receive it.
- **Submit the results through Greenhouse in a ZIP file.**
- **Your deliverable must be self-contained**, so please include any files that are needed to run your process, use your notebooks, etc.

What we value:

- Following the instructions and doing everything from the checklist
- Clear, coherent and concise written communication
- Clean coding style
- Efficient solutions to the problems given

1. KPIs (20 points)

OUTPUT FORMAT	● A **PDF** file
WHAT TO DO **[Checklist]**	□ In your opinion, **what are three important KPIs** for Glovo and why? ***Note:*** *Ignore pure financial KPIs that apply to every business.* Please **choose one** of these three KPIs and: □ Make an educated guess of its value and provide a step-by-step explanation of your guess. □ Suggest at least one idea that could significantly improve this KPI.
WHAT ARE WE LOOKING FOR?	● Good understanding of the business challenges we are facing ● Ability to **communicate and summarize**

2. SQL (40 points)

OUTPUT FORMAT	DB Fiddle link that actually runs on **PostgreSQL 9.6** OR SQL Fiddle link that actually runs on **PostgreSQL 9.6**.
WHAT TO DO **[Checklist]**	□ DDL for creating the SQL tables: `customer_courier_chat_messages` and `orders` □ Insert data in the SQL tables (you can use the one from the example below) □ Create the final table with a query □ [OPTIONAL] Include tests for the final table
WHAT ARE WE LOOKING FOR?	● Fundamental **SQL knowledge** ● Ability to write clean, understandable and efficient queries

You have the ***customer_courier_chat_messages*** table that stores data about individual messages exchanged between customers and couriers via the in-app chat. An example of the table is below:

Sender app type	Customer id	From id	To id	Chat started by message	Order id	Order stage	Courier id	Message sent time
Customer iOS	17071099	17071099	16293039	FALSE	59528555	PICKING_UP	16293039	2019-08-19 8:01:47
Courier iOS	17071099	16293039	17071099	FALSE	59528555	ARRIVING	16293039	2019-08-19 8:01:04
Customer iOS	17071099	17071099	16293039	FALSE	59528555	PICKING_UP	16293039	2019-08-19 8:00:04
Courier Android	12874122	18325287	12874122	TRUE	59528038	ADDRESS_DELIVERY	18325287	2019-08-19 7:59:33

You also have access to the **orders** table where you have an order_id and city_code field.

Your task is to build a query that creates a table (including the DDL CREATE statement) (***customer_courier_conversations***) that aggregates individual messages into conversations. Take into consideration that a conversation is unique per order. The required fields are the following:

- **order_id**
- **city_code**
- **first_courier_message**: Timestamp of the first courier message
- **first_customer_message**: Timestamp of the first customer message
- **num_messages_courier**: Number of messages sent by courier
- **num_messages_customer**: Number of messages sent by customer
- **first_message_by**: The first message sender (courier or customer)
- **conversation_started_at**: Timestamp of the first message in the conversation
- **first_responsetime_delay_seconds**: Time (in secs) elapsed until the first message was responded
- **last_message_time**: Timestamp of the last message sent
- **last_message_order_stage**: The stage of the order when the last message was sent

Make your query scalable and readable!

3. Events Processing (40 points)

OUTPUT FORMAT	A ZIP file containing: • The code in a **Python file / Jupyter notebook** • The **input data** • The **output data** as a **single CSV file** • [OPTIONAL] Include the config files (like requirements.txt or pyproject.toml)
WHAT TO DO **[Checklist]**	□ **Develop ETL code for creating** the output. We suggest you to use Pandas or Spark □ Declare **functions** with docstrings □ Include **comments** so that it’s easy to follow □ Write **clean code** following Python standards
WHAT ARE WE	• Proficiency in **Python** and related data processing packages

LOOKING FOR?	(Pandas/PySpark) ● Following coding best practices

You have been given a dataset (*appEventProcessingDataset.tar.gz*) of three csv files containing the following:

1. Client HTTP endpoint polling event data for a set of devices running a web application.
2. Internet connectivity status logs for the above set of devices, generated when a device goes offline whilst running the application.
3. Orders data for orders that have been dispatched to devices running the above web application.

We're interested in knowing about the connectivity environment of a device in the period of time surrounding when an order is dispatched to it.

Using Python and any useful libraries, ***produce a single csv formatted output dataset*** that contains the following information:
For each order dispatched to a device:

- The total count of all polling events
- The count of each type of polling status_code
- The count of each type of polling error_code and the count of responses without error codes.

...across the following periods of time:

- Three minutes before the order creation time
- Three minutes after the order creation time
- One hour before the order creation time

In addition to the above, across an unbounded period of time, we would like to know:

1. The time of the polling event immediately preceding, and immediately following the order creation time.

2. The most recent connectivity status (“ONLINE” or “OFFLINE”) before an order, and at what time the order changed to this status. This can be across any period of time before the order creation time. Not all devices have a connectivity status.

Please include all code used to produce your output, and any explanatory notes.