Technical test

Welcome to the Data Engineering technical challenge, read the document before starting.

General Requirements:

- Candidate must use Python3
- Store data in a csv (1 csv per table) except the lake

Guidelines:

- Push code to your Github account and share the link to your answer via email
- Readable code
- Commented code
- Documentation on how to execute the scripts or any docker implementation
- Feel free to ask questions to mariana.velasco@bitso.com

Bonus:

- Unit testing
- Error handling
- PEP8 comment style

Challenge 1

“Markets team” needs to monitor the bid-ask spread from the “order books” MXN_BTC and USD_MXN, they need to do custom analysis on the bid-ask spread and create alerts whenever the spread is bigger than 1.0%, 0.5%, 0.1% or any other custom value in one second observations.

To enable them you need to create a process that gets the order books each second and extract the necessary values to power the analysis. The process should store a file every 10 min with all the observations gathered (600 records) in that timespan. This file will be stored in the “data lake” and must be partitioned (You’ll define the relevant partitions).

Details:

- Spread = (best_ask - best_bid) * 100/ best_ask
- Use bitso api
- Create a record with the spread of the orderbook every second
    - The table should have the following schema (orderbook_timestamp: string, book: string, bid :float, ask: float, spread: float)
    - Ex. (“2022-04-30T04:10:26+00:00”, “btc_mxn”, 790333.41, 790961.89,0.079)
- The code must generate a directory structure that simulates the partitions in s3

Deliverables:

- Python 3 source code with your solution
- Output files of your solution
- read.me file justifying the reason you choose those partition keys

Bonus:

- Bonus points if you use webhooks
- Main script orchestrating the pipeline or the use airflow

Challenge 2

In this URL you'll find 4 csv's, each csv is a snapshot of the tables deposit, withdrawal, event and user with historic information. As a data engineer you need to provide master data for our downstream users Business Intelligence, Machine Learning, Experimentation and Marketing. These teams will use the data for multiple purposes, the tables and data model you'll create should help them answer questions like the following:

- How many users were active on a given day (they made a deposit or withdrawal)
- Identify users haven't made a deposit
- Identify on a given day which users have made more than 5 deposits historically
- When was the last time a user made a login
- How many times a user has made a login between two dates
- Number of unique currencies deposited on a given day
- Number of unique currencies withdrew on a given day
- Total amount deposited of a given currency on a given day

Details:

- Use the data modeling technique that you consider appropriate for the use cases mentioned above. We expect an advanced approach to ensure the highest quality of data modeling.
- Code your ETL and write the output tables in CSVs files, one per table.
- Create at least 4 queries that will answer 4 different use cases with your solution and store it on an sql or txt file

Deliverables:

- Image with the ERD of the data model
- Python 3 source code with your solution
- CSVs with the output tables
- SQL or TXT file with the queries that will answer at least 3 cases
- read.me file explaining what modeling techniques did you use, why did you choose them and what would be potential downside of this approach

Bonus:

- ETL process daily batches
- Main script orchestrating the pipeline or the use airflow
- Implement Unit testing