BLUE BOTTLE
COFFEE

Data Engineer Coding Challenge

Justin Flick

I. Approach

My goal for this solution was to demonstrate my ability to write clean, fast, and modular code. These 3 principles informed my choice of an object-oriented design pattern, as I felt it would allow for maximum modularity. At a more granular level, I chose to base my code around the Pandas library. Pandas is an abstraction on top of the NumPy library which directly interfaces with C-level APIs to offer vastly improved performance compared to raw Python.

I chose to use a SQLite database to avoid any complications with database access and/or credentialing when the Blue Bottle team runs my code. However, in a real production environment, I would typically reach for a cloud-based database to abstract away database administration tasks and to offer the accessibility and scalability required of a production-level database.

Performance heavily informed my design choices. I ended up reaching for the Dask library to allow for parallel processing of my data transformation activities. This library made it easy to partition my dataset into smaller chunks that could then be processed concurrently.

I wanted to make my code design very straightforward. As such, I wrote a series of granular extract, clean, and load methods to allow for clear visibility into the ETL pipeline. Once I began to develop reports, I chose to make each report it’s own monolithic method, in order to allow for easy repeatability, since reports will commonly be run multiple times. Ultimately, I’m proud of the solution you see before you.

II. Queries

Report 1:

```
SELECT temp as 'Temperature (Whole Degress Farenheit)',
item_name as 'Item Name',
MAX(Number_Sold) as 'Number Sold'
FROM (SELECT temp,
item_name as Item_Name,
SUM(net_quantity) as Number_Sold
FROM Sales
GROUP BY temp, item_name
ORDER BY temp, Number_Sold DESC)
GROUP BY temp
ORDER BY temp ASC
```

Report 2:

```
SELECT strftime('%Y-%m-%d', local_created_at),
item_name, net_quantity,
temp FROM Sales
GROUP BY item_name, strftime('%Y-%m-%d', local_created_at)
ORDER BY  strftime('%Y-%m-%d', local_created_at), item_name DESC"""
```

```
SELECT c.item_name as 'Item Name', c.avg_change_cold as 'Average Change
in Sales when Colder' , w.avg_change_warm as 'Average Change in Sales
when Warmer'
FROM (SELECT item_name, AVG(Change_in_sales) as avg_change_cold FROM
Sales_Trends WHERE Changes_in_temp = -2 GROUP BY item_name) as c
INNER JOIN (SELECT item_name, AVG(Change_in_sales) as avg_change_warm
FROM Sales_Trends WHERE Changes_in_temp = 2 GROUP BY item_name) as w ON
c.item_name = w.item_name
```

BLUE BOTTLE
COFFEE

III. Instructions to Run Code

NOTE Code was tested on a Windows 10 system. This code requires Python 3.6 or higher

Step	Description
1.	Clone the following repository: https://github.com/Jflick58/Blue_Bottle_Coding_Challenge.git
2.	Navigate to the location you cloned the repo and open a command prompt

3.	Run the following command: `pip install -r requirements.txt`
	NOTE I recommend using a virtual environment, to set this up in a virtual environment run the following commands:
	`Virtualenv venv`
	`venv\Scripts\activate`
	Then do step 3

4.	Add morse.csv to your directory
5.	Open challenge.py in your text editor. In the API_KEY global variable, replace 'your_key_here' with [redacted] and save the file.
6.	By default, the program only runs the reports. To change this, uncomment `#run_entire_pipeline()` below `if __name__ == '__main__':` `and` save the file.
7.	In your command prompt run the following command: `Python challenge.py` The code should run. Once finished, report_1.csv and report_2.csv will be overwritten in your directory

BLUE BOTTLE
COFFEE

IV. Extra Credit

How would you productize both reports?

First, I would focus on automating the collection of sales data. In an ideal world, I would build a real-time pipeline from Blue Bottle's point-of-sale systems using something like Google Cloud Pub/Sub or Amazon Kinesis. I'd then rewrite the ETL using a tool like Apache Spark or Beam. If we were to expand these reports to other cafes, I would likely choose to utilize a Hadoop Cluster or Google BigQuery to effectively cluster and scale the vast amount of data we would be dealing with.

Second, I would like to focus on creating a better presentation format for the data. I would love to use Tableau or a custom Chart.js-based website to build out automatically updating and interactive visualizations and reports. We could then integrate that into an existing reporting portal or use it as a catalyst to build out a centralized analytics portal for internal business intelligence. I'd also like to build a small function to allow users to subscribe to these reports and receive them automatically via email.

From a backfill perspective, once we have the initial automation built, I'd like to build out the proper queries to pull legacy data out of Blue Bottle's existing data stores, format it for use in this approach, and upload to the new datastore.

What are some tradeoffs and assumptions for your design of this ETL?

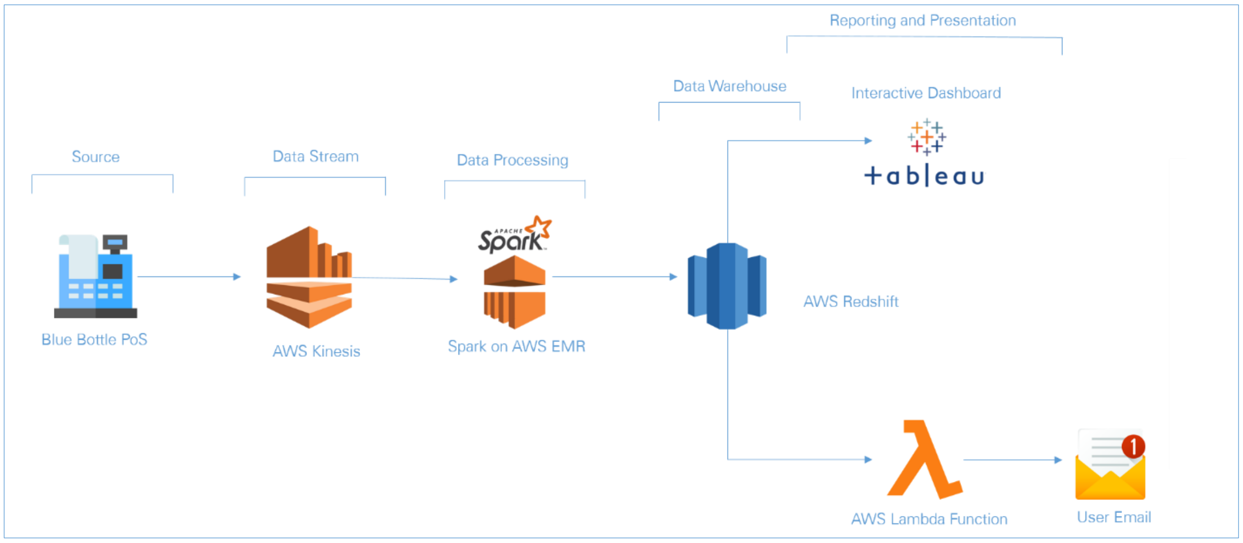
For the current design, I primarily had to tradeoff performance for reduced development time. As I was limited on time to work on this task due to my obligations to my current employer, I didn't have the time to spend finely tuning the performance of the ETL pipeline. In the end, I still was able to speed up the processing time significantly by using the Dask library which allowed me to easily partition and run the ETL process on the dataset in parallel. I find this to be an example of effectively managing tradeoffs; while I would have liked to implement PyPy or Cython or Asynchronous HTTP requests, I still was able to make meaningful performance gains while minimizing the effects of the tradeoff. I think I also ended up trading off some performance on the second report by using an intermittent dataframe to generate calculated fields rather than pure SQL, but since I was having issues calculating the fields I felt that I was still able to deliver solid report generation performance.

In terms of assumptions made during the development process of this ETL, I had to make some assumptions about the goal of the report. For example, I managed to pull in hourly weather data, but the second report inquired about temperature changes over the course of days. Ultimately, I made the assumption that days as the unit of time was correct, and built the report around that. Additionally, I wasn't sure if the goal was to measure the change in sales when the temperature specifically changes two degrees, or if the report was intended to measure sales variance when the temperature changes at least 2 degrees. I again chose to make the assumption that the requestor of the report wanted a specific temperature measure, and used specific values.

From a more general perspective, I made the assumption that there were no restrictions on dependency choice, as I did end up using multiple open source libraries in this coding challenge. Ultimately, given the tradeoffs I encountered and the assumptions I was forced to make, I feel like I delivered a solid application.

What some of the tools you would consider to build this into an ETL pipeline?

As I mentioned before, I would really like to turn this into a modern, highly-scalable, and real-time data pipeline. As I have had quite a bit of experience with various cloud platforms, I would tend to reach for some of the offerings of the major cloud providers. For example, I could see a good stack looking like this:



BLUE BOTTLE
COFFEE