Approach and walkthrough

When approaching a given dataset to be ingested, I generally like to ask myself a series of question of the data I am to be working with. This list is adapted from a similar set of questions from an old coworker of mine at a previous job.

Let's briefly discuss a few of these main questions:

What is the data?
- What data are stored in each row?
- Is the data overly ambiguous or hard to understand?
- What is the level of detail?

Why is the data saved?
- What problem is the data solving or what questions is it providing an answer to?
- Is the problem being answered a repeated query? A one-off analysis?

Where is the data from?
- What assumptions and biases are built into the data?
- How has the data been treated before ingestion?

What could go wrong?
- How might the pipeline break from an upstream change?
- Could there be a change in scale of data between batches?
- Are we future-proofing for format and datatype changes?
- Implement issue detection to actively catch changes as soon as they occur
- Communicate!

Assuming the dataset ingestion pipeline has been properly implemented, we can discuss the cleaning process using the example F&B dataset:

Big questions:

- What are we looking at?

Each row is one order_id (with some duplicate rows that we remove easily) that contains details like location (vendor), product name and type, date time, and event name and season. In many cases, there are also columns that are duplicates, or just not needed for the dataset. For example, this dataset is exclusively from BMO stadium and payment type is always the same value, so we could consider excluding those columns in our ingestion and keep those pieces of metadata in the pipeline tags. More simply, in the case of two identical order_id columns, we can immediately drop the duplicate without a second thought.

- Data types and shape

Automated data type interpretation in Pandas can be memory intensive. So, if needed, manual datatype assignment could be used to avoid such issues and is commonplace in most data ingestion pipelines. This can also help prevent major errors and breaks in the pipeline due to changes in data types upstream.

- Nested data like lists of lists, dicts, and JSON

Nested data can be a great way to store richer data like custom web event attributes without needing hundreds of different, overly-specific columns to store them in. In this dataset, we could consider a case where we want to aggregate all vendor_ids that sell certain products in order to build a model of ideal concessions and vendor coverage for each seating area in the stadium.

- What data is missing?

Not all Null values are created equal. Root causes may include data joins, human input error, PoS system issues, or even just missing values, among many many other reasons.

Constructing a basic correlation table from True/False values, we can examine the frequency at which different column values are Null at the same time. This can help guide us towards understanding our data better: as seen above, total and subtotal are Null with order_id at the same rate, implying that those two fields are very likely Null when the other is Null.

```
In [18]: data.isna().corr()
```

Out[18]:

	order_id	name	price	type	id	quantity	tax	total
order_id	1.000000	NaN	NaN	-0.000064	NaN	NaN	NaN	0.007377
name	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
price	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
type	-0.000064	NaN	NaN	1.000000	NaN	NaN	NaN	0.001835
id	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
quantity	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
tax	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total	0.007377	NaN	NaN	0.001835	NaN	NaN	NaN	1.000000
original_subtotal	0.007377	NaN	NaN	0.001835	NaN	NaN	NaN	1.000000
status	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
vendor_id	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
vendor_name	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
venue_name	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- How can we fix Null values?

The unfortunate truth is that sometimes we can't, but understanding where the missing data originates from is often more important than fixing the ones currently missing so the problem can be resolved. I have used some techniques on the example dataset to try to fill in the gaps. For example, to backfill some missing Type values, I created a mapping dict using the most frequent Type value for the given Id value:

```
In [49]: id_to_type_map = dict(data.groupby('id')['type'].agg(pd.Series.mode))
```

While there are Id values with multiple Type values, this method is a quick (and slightly dirty) version of a comprehensive mapping table for Id values. Ideally, this data would be stored and maintained in a separate location and joined to the data as needed, potentially eliminating the need for Type to be ingested along with the rest of the data source in order to reduce read/write time.

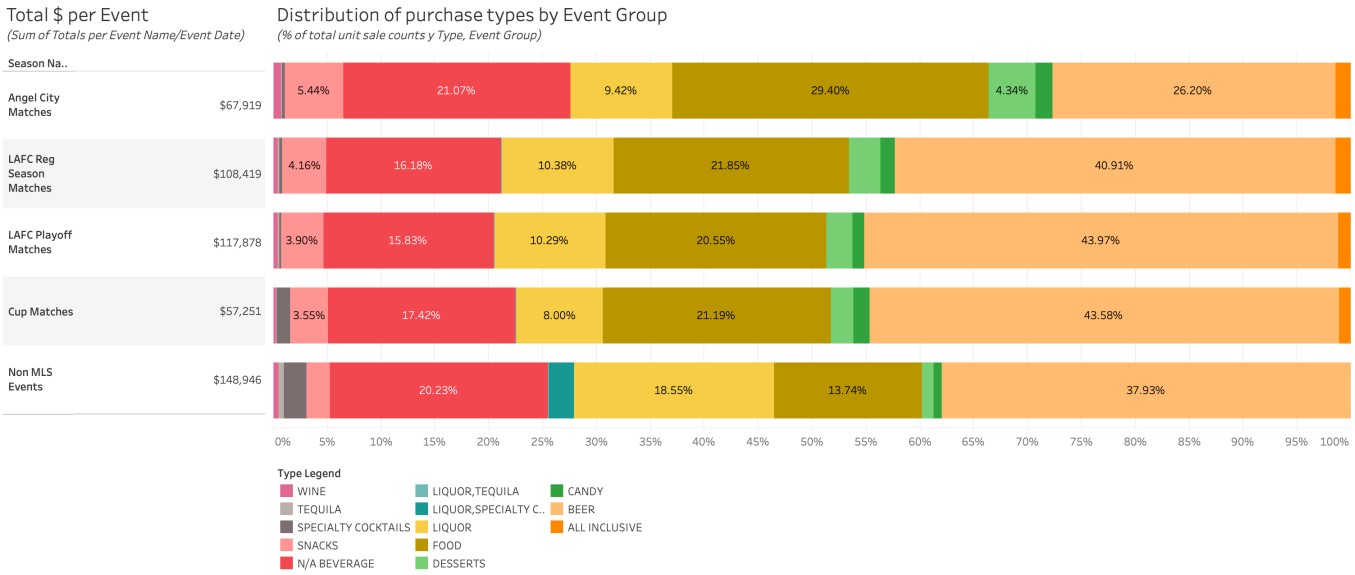
Seen to the right are examples of Id values appended with SPEED BAR, where the same product name might have different Type values. Assigning the mode of these values to the dict mapping table will allow us to approximate the most applicable value, but this approach would need refinement in an actual production database.

```
In [40]: data[data.type.str.find(',')>0].type
```

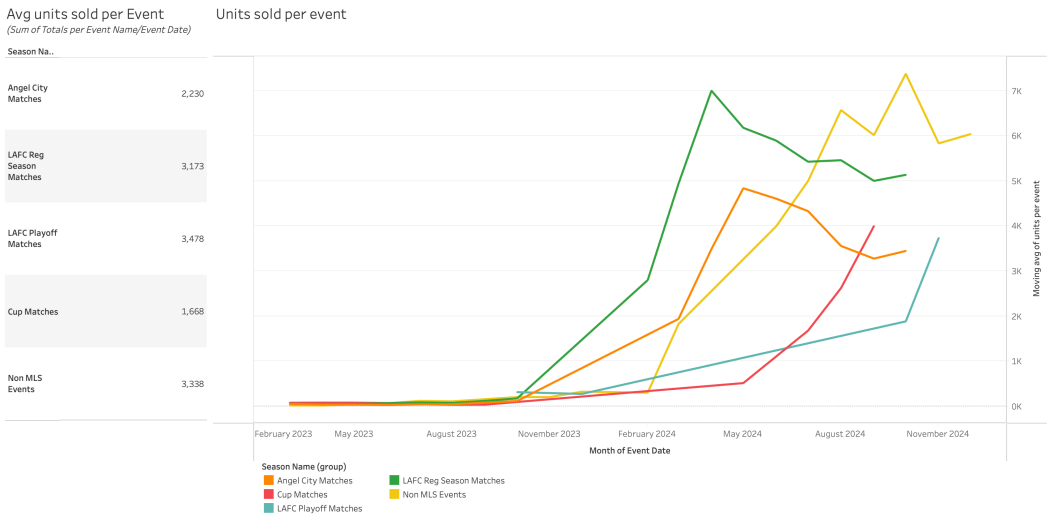
Out[40]:

```
8                      BEER,SPEED BAR
10                     LIQUOR,TEQUILA
12          LIQUOR,SPECIALTY COCKTAILS
17                     FOOD,SPEED BAR
27                     BEER,SPEED BAR
                          ...
377411                 FOOD,SPEED BAR
377423                 BEER,SPEED BAR
377426       LIQUOR,SPEED BAR,TEQUILA
377441       LIQUOR,SPEED BAR,TEQUILA
377446      LIQUOR,SPECIALTY COCKTAILS
Name: type, Length: 16651, dtype: object
```

Once we are happy with the cleanliness of the data, we are free to do some introductory analysis and visualization on the dataset. In these two graphics, we can quickly understand the differences in buying habits of the average audiences for different event groups and average Total $ per event.



Understanding the proclivities of each distinct group can enable us to make better decisions about how best to incentivize purchasing through deals and introducing new products. Lastly, we have a similar layout to help us monitor how unit sales are for these same event groups. We could additionally create a dashboard specifically for WoW/MoM metrics to provide in-depth trend analytics.



To finalize, it's worth mentioning ways we could dive deeper into data. One of the best ways to enhance this F&B dataset would be to have access to ticketing data to look at similar metrics per ticket holder to normalize for attendance and examine vendor hotspots using seat location.