Actable Schema Technical Documentation

Client data is incredibly diverse and sophisticated. In order to create greater efficiency in transforming client data into meaningful insights and feature sets, Actable has built a `standard schema`. An example of the `standard schema`, including descriptions of its attributes, is detailed below. The client will either transform their data or will leverage Actable resources into this format. The Actable `standard schema` provides the following advantages:

- A long-data format that Actable data ingestion processes can immediately synthesize to **extract insights** and prepare for modeling
- Long format allows Actable armature to be language-agnostic, retaining original client nomenclature and data labeling
- Flexible grouping categories allows for critical segmentation for clients' unique group types
- Customer includes data of interest in relevant data categorization
- `event_dt` opens up scale of analysis to deliver key performance indicators across ids, metrics, and groups

The final goal of this is to massively increase production of insights reports and generate feature sets for machine learning application.

Schema Classes

These are four different types of data that represent a unique component of the data corpus. Each can be labeled with numbers and letters to include greater complexity within the data.

1. `id`
2. `metric`
3. `group`
4. `event_dt`

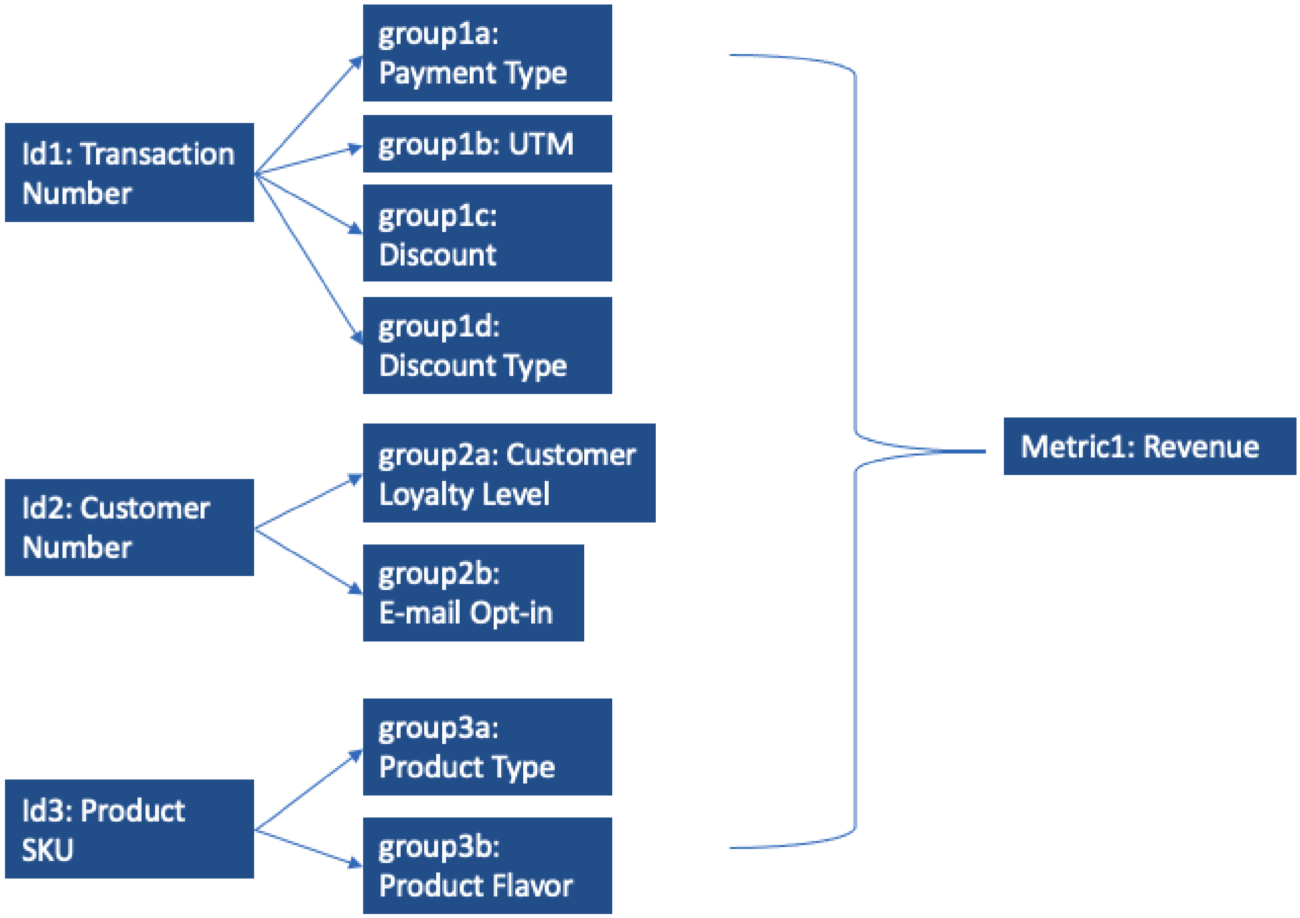
The numbers and letters across the four `classes` indicate that the columns have a relationship with each other, and are called `binders`. For example, `id1` is associated with `metric1`, `group1a` , `group1b`, `group1c`, and `group1d`.

Example

Below is a sample table that includes all of these categories:

	id1	id2	id3	metric1	group1a	group1b	group1c	group1d	group2a	group2b	group3a	group3b	event_dt
1	451-414	13573	209	10.42	credit	paid_search	0.1	product_discount	NULL	1	chews	sour	1/13/20
2	124-123	13621	474	3.85	gift card	paid_search	0.4	order_discount	silver	1	lollipops	special	1/6/20
3	124-123	13621	98	12.42	credit	paid_search	0.4	order_discount	silver	1	bites	regular	1/17/20
4	254-673	14096	239	1.57	gift card	email	0.3	product_discount	platinum	0	lollipops	sour	1/15/20
5	165-165	12952	450	11.57	gift card	org_search	0.2	order_discount	bronze	0	hard candy	regular	1/8/20

Map of class interactions with binders



Each client's `standard schema` will have a corresponding `schema dictionary` that serves as the data dictionary, allowing Actable to leverage custom client nomenclature. Here is an example of a `schema dictionary`:

	schema_colname	schema_colnumber	description
1	transaction_number	1	transaction identifier (appears once per product per transaction)
2	customer_id	2	customer identifier (unique per customer)
3	product_id	3	product identifier (unique per SKU)
4	revenue	4	revenue associated with each line item
5	transaction_type	5	credit or gift card order
6	utm	6	traffic source
7	discount	7	discount amount transaction or product
8	discount_type	8	type of discount – apply to sum of transaction revenue for order_discount and directly to product for product_discount
9	customer_loyalty	9	loyalty program level
10	email_optin	10	email opt-in flag
11	product_type	11	classification of SKU
12	flavor	12	another classification of SKU
13	event_dt	13	date of transaction

Data Classes

id

id1	id2	id3
451-414	13573	209
124-123	13621	474
124-123	13621	98
254-673	14096	239
165-165	12952	450g

`ids`, simply put, are our primary identifiers. In mathematical terms, anything that can be considered the denominator of an analysis should be included in the `id` section.

The most granular `id` will always be labeled `id1` , and will scale up per individual `binder` level. This is to allow the client to provide far greater detail than relying on one specific `id` column. For instance, a common data structure in e-commerce is the presence of both a `transaction_number` (representing the specific transaction that occurred) and a `customer_id` (representing the specific customer who made the purchase). Both represent important aspects of any insights project, and both are integral to these key performance indicators:

- Average Life Time Value (LTV): the total spend of a customer
- Average Order Value (AOV): the average spend per transaction

Average LTV =

```
sum(total revenue) / total number of customers
```

AOV =

```
sum(total revenue) / total number of transactions
```

We cannot obtain LTV without a unique customer identifier, but we also cannot obtain AOV without a unique transaction identifier. Including both allows for flexible analysis. Additionally, in our example, another `id` is required to identify the type of product being purchased in each transaction. This allows for product level analysis.

In practical terms, the `id` section can scale out as far as the client wishes, assuming it provides a unique ‘base’ of analysis.

metric

metric1
10.42
3.85
12.42
1.57
11.57

`metrics` are any numeric value that represents the size or scale of the transaction (or customer interaction) with the client. Most of the time, `metric1` will be revenue or cost of a transaction / sub-transaction. Other examples of metrics could be quantity of a product purchased, number of page views, or number of clicks. The `binder` serves to indicate which `id` the `metric` is associated with – ie, revenue associated with a transaction. `metric` is very straightforward; it provides a way to measure individual `ids`. Any number of `metric` columns can be included for analysis.

group

group1a	group1b	group1c	group1d	group2a	group2b	group3a	group3b
credit	paid_search	0.1	product_discount	NULL	1	chews	sour
gift card	paid_search	0.4	order_discount	silver	1	lollipops	special
credit	paid_search	0.4	order_discount	silver	1	bites	regular
gift card	email	0.3	product_discount	platinum	0	lollipops	sour
gift card	org_search	0.2	order_discount	bronze	0	hard candy	regular

If id is the unique identifier, and metrics are representation of size of transaction or quantity, groups are the filtering that allow for segmentation and comparison.

group allows for subsetting of all ids into relevant segmentations. Each group should represent a clear subset of the id that shares a binder. In the example above, group1a represents the different payment types available on a transaction, group2a represents different customer loyalty levels associated with the customer id, group3a represents different product types matched to the SKU.

groups allow aggregation to occur across segments by any id, calculating metric statistics for any column that shares the same binder. Additionally, some ids can be aggregated by all groups, regardless of binder label. For instance, in the above example, id2 (customer_id) can be used to calculate the number of customers who purchased a 'hard candy' product from group3a. This is why it is important for the most granular ids to be first – this allows for statistics to be calculated across groups with different binder identifiers.

event_dt

event_dt
1/13/20
1/6/20
1/17/20
1/15/20
1/8/20

event_dt provides a datetime stamp for relevant event occurrence – this is typically associated with the most granular identifier, id1. event_dt answers 'when did this event (id1) occur?'

This column allows for calculation of all of the above subset by both date time and by date period, and can be applied on all other columns.

Sample data points generated from event_dt:

- average time between purchases
- purchase frequency
- time since last purchase
- day of week of purchase
- weekend flag