This document outlines the analysis on Amazon.Listings dataset for Sprint 3.

Feedback from last meeting with Payability:

1. Group listings by mp_sup_key and asin value as one listing appears several times due to the partition_date.
2. Create a time series data visualization to show the change in Inventory data based on the quantity and price.
3. Resolve query issues from Sprint 2.
4. Focus on building a dashboard to show listing behavior for a particular seller (micro view) and how they compare to the rest of the sellers.

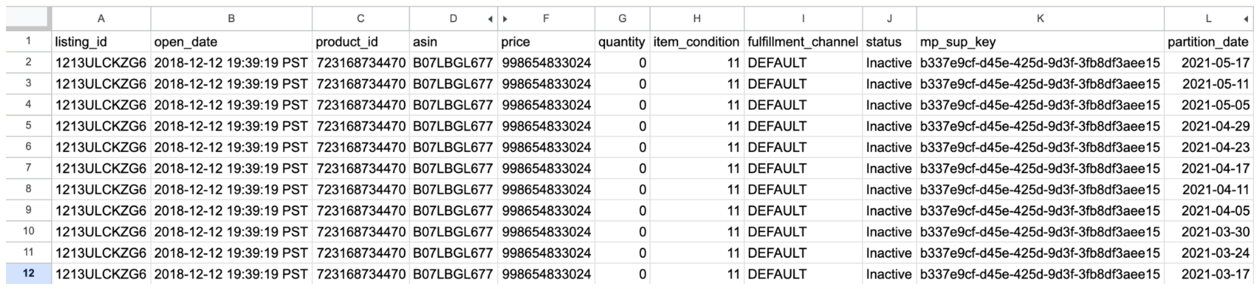
Calculating Inventory value for sellers:

Data Duplication issue:

Because `998654833023.95` was an unusually high price point for one listing, I wanted to look into it and analyze the difference between the rows returned by the sql query. The result was 92 rows of data where all the columns had the same value except the partition date. This confirms my hypothesis that the same data gets sent through the api for every partition.

```sql
SELECT *
FROM `bigqueryexport-183608.amazon.listings`
where price = '998654833023.95'
order by partition_date desc;
```

This is a sample of the results from the google sheets. Data is ordered in descending order by the partition_date.

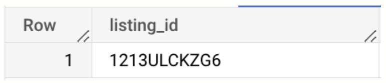


	A	B	C	D	F	G	H	I	J	K	L
1	listing_id	open_date	product_id	asin	price	quantity	item_condition	fulfillment_channel	status	mp_sup_key	partition_date
2	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-05-17
3	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-05-11
4	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-05-05
5	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-04-29
6	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-04-23
7	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-04-17
8	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-04-11
9	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-04-05
10	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-03-30
11	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-03-24
12	1213ULCKZG6	2018-12-12 19:39:19 PST	723168734470	B07LBGL677	998654833024	0	11	DEFAULT	Inactive	b337e9cf-d45e-425d-9d3f-3fb8df3aee15	2021-03-17

When querying for a unique listing_id, the result is only one id.

```sql
SELECT DISTINCT listing_id
```

```
FROM (SELECT * FROM `bigqueryexport-183608.amazon.listings`
where price = '998654833023.95');
```

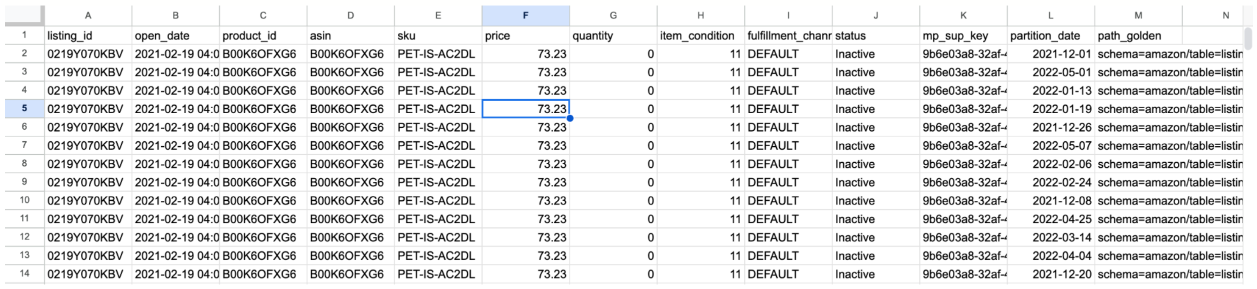


Row	listing_id
1	1213ULCKZG6

This means that omitting the partition key could return unique listings only.

Another way to verify: for example for listing_id `'0219Y070KBV',` there are 29 rows of results. This is again because of the value on the partition_date column.

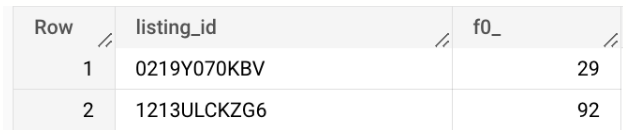
```
SELECT *
FROM `bigqueryexport-183608.amazon.listings`
WHERE listing_id = '0219Y070KBV';
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	listing_id	open_date	product_id	asin	sku	price	quantity	item_condition	fulfillment_chanr	status	mp_sup_key	partition_date	path_golden	
2	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2021-12-01	schema=amazon/table=listin	
3	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-05-01	schema=amazon/table=listin	
4	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-01-13	schema=amazon/table=listin	
5	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-01-19	schema=amazon/table=listin	
6	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2021-12-26	schema=amazon/table=listin	
7	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-05-07	schema=amazon/table=listin	
8	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-02-06	schema=amazon/table=listin	
9	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-02-24	schema=amazon/table=listin	
10	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2021-12-08	schema=amazon/table=listin	
11	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-04-25	schema=amazon/table=listin	
12	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-03-14	schema=amazon/table=listin	
13	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2022-04-04	schema=amazon/table=listin	
14	0219Y070KBV	2021-02-19 04:0	B00K6OFXG6	B00K6OFXG6	PET-IS-AC2DL	73.23	0	11	DEFAULT	Inactive	9b6e03a8-32af-4	2021-12-20	schema=amazon/table=listin	

The results of this query confirms my hypothesis.

```
select distinct listing_id, count(*)
FROM `bigqueryexport-183608.amazon.listings`
WHERE listing_id in ('0219Y070KBV','1213ULCKZG6')
group by listing_id;
```



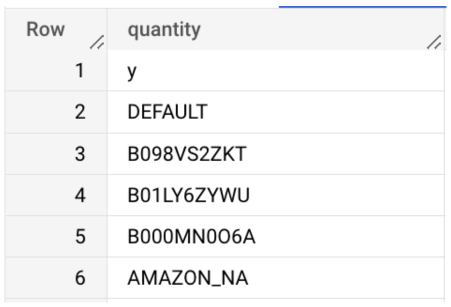
Row	listing_id	f0_
1	0219Y070KBV	29
2	1213ULCKZG6	92

Quantity issue:

This is a string nullable field in the dataset. To calculate the inventory value (i.e. quantity x price), this has to be converted to a float.

When looking at the values in this field, I find two issues:

1. *String fields that are not numbers:* In my assumption, #3 - 5 are either product_id or asin values. #2 and #6 are from the fulfillment_channel field. These values are irrelevant to the context of quantity.

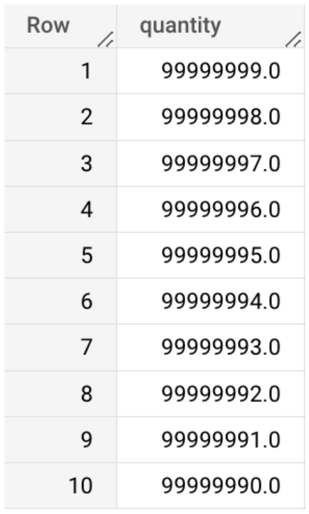


Row	quantity
1	y
2	DEFAULT
3	B098VS2ZKT
4	B01LY6ZYWU
5	B000MN0O6A
6	AMAZON_NA

```
SELECT DISTINCT quantity
FROM `bigqueryexport-183608.amazon.listings`
order by quantity desc;
```

2. *Very large numbers:* Some numbers in the quantity field seem unreasonably large. My assumption is that these are dummy numbers and not related to an active listing on Amazon marketplace.

```
SELECT DISTINCT IFNULL(SAFE_CAST(quantity AS FLOAT64), 0.0) AS quantity
FROM `bigqueryexport-183608.amazon.listings`
order by quantity desc;
```

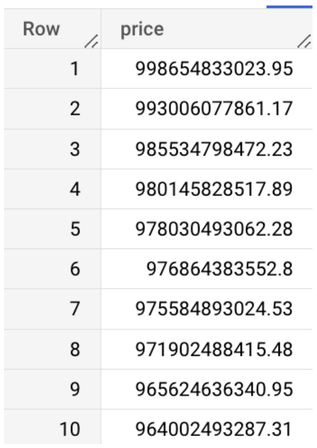


Row	quantity
1	99999999.0
2	99999998.0
3	99999997.0
4	99999996.0
5	99999995.0
6	99999994.0
7	99999993.0
8	99999992.0
9	99999991.0
10	99999990.0

Price issue:

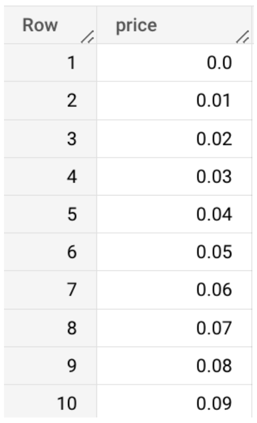
Similar to quantity, when we look at the values in descending order, we can see some very big numbers. It does not seem plausible for the price of a product listing to be these numbers.

```
SELECT DISTINCT IFNULL(SAFE_CAST(price AS FLOAT64), 0.0) AS price
FROM `bigqueryexport-183608.amazon.listings`
order by price desc;
```



Row	price
1	998654833023.95
2	993006077861.17
3	985534798472.23
4	980145828517.89
5	978030493062.28
6	976864383552.8
7	975584893024.53
8	971902488415.48
9	965624636340.95
10	964002493287.31

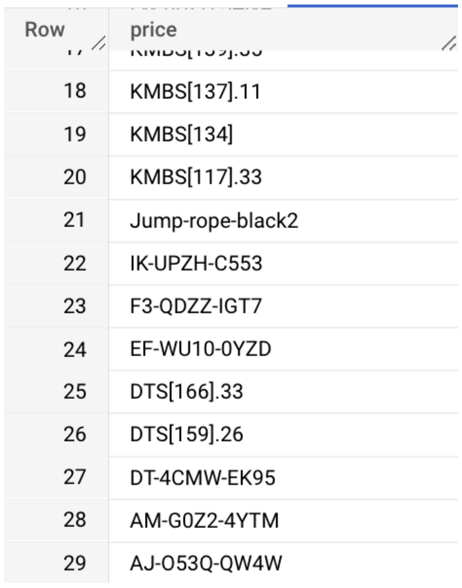
Similarly, ordering by ascending order also shows some weird values for the price column. These values are very small to be the price of a listing.



Row	price
1	0.0
2	0.01
3	0.02
4	0.03
5	0.04
6	0.05
7	0.06
8	0.07
9	0.08
10	0.09

To calculate the price of a listing we have to take an average over the price of all the listings under the same listing_id. Doing so without removing these values will give false results.

Before converting to float, we can notice that the price field also has bad values. For example:



Row	price
18	KMBS[137].11
19	KMBS[134]
20	KMBS[117].33
21	Jump-rope-black2
22	IK-UPZH-C553
23	F3-QDZZ-IGT7
24	EF-WU10-0YZD
25	DTS[166].33
26	DTS[159].26
27	DT-4CMW-EK95
28	AM-G0Z2-4YTM
29	AJ-O53Q-QW4W

<u>Calculating total no. of Listings based on fulfillment_channels:</u>

I am limiting my query to only include three types of fulfillment channels as based on the analysis in Sprint 2, these were the top 3 values that were present in more than 90% of the data.

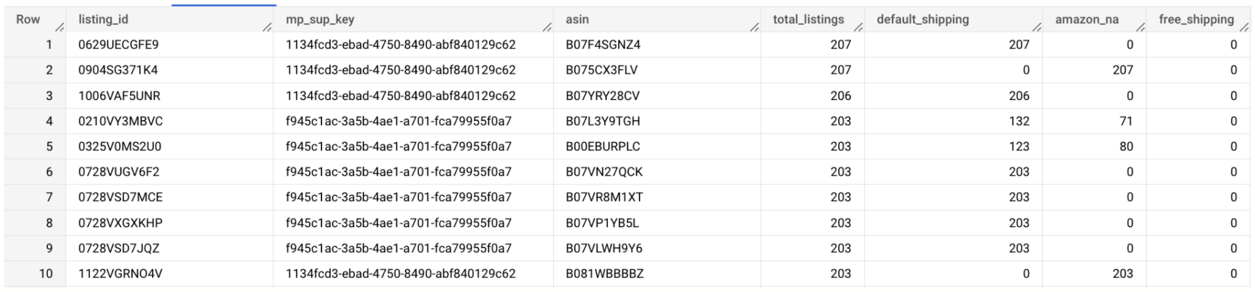
```
SELECT
 listing_id,
 mp_sup_key,
 asin,
 COUNT(*) as total_listings,
 COUNT(IF(fulfillment_channel='DEFAULT', 1, NULL)) as default_shipping,
 COUNT(IF(fulfillment_channel='AMAZON_NA', 1, NULL)) as amazon_na,
 COUNT(IF(fulfillment_channel='FREE SHIPPING', 1, NULL)) as free_shipping,
FROM
 `bigqueryexport-183608.amazon.listings`
WHERE
 mp_sup_key IS NOT NULL
 AND asin IS NOT NULL
 AND listing_id IS NOT NULL
 AND listing_id <> ''
```

```
GROUP BY
 listing_id,
 mp_sup_key,
 asin
ORDER BY total_listings desc;
```

The distribution of fulfillment channels among all the records under a listing_id is captured in this dashboard. This dashboard is built using 2021 data only. We can see that out of all the listings that were opened in 2021, only 7.5% of fulfillment was done through Amazon; remaining were fulfilled through DEFAULT channel. Two issues with this conclusion:

1. This is contrary to what SMEs in Payability believe to be true. Based on their knowledge, a higher number of customers use Amazon fulfillment services to sell items on their marketplace.
2. What does the DEFAULT channel mean?

Also, the result of this query challenges my original assumption about a listing_id. While we see that for one listing_id, there are more than one record available, we also can see from this table that the fulfillment_channel value for some of these records vary. For example line no. 5, we can see that for the same listing of a product by one seller, out of 203 records, 123 were fulfilled using default_shipping method and remaining 80 were fulfilled using amazon_na.



Row	listing_id	mp_sup_key	asin	total_listings	default_shipping	amazon_na	free_shipping
1	0629UECGFE9	1134fcd3-ebad-4750-8490-abf840129c62	B07F4SGNZ4	207	207	0	0
2	0904SG371K4	1134fcd3-ebad-4750-8490-abf840129c62	B075CX3FLV	207	0	207	0
3	1006VAF5UNR	1134fcd3-ebad-4750-8490-abf840129c62	B07YRY28CV	206	206	0	0
4	0210VY3MBVC	f945c1ac-3a5b-4ae1-a701-fca79955f0a7	B07L3Y9TGH	203	132	71	0
5	0325V0MS2U0	f945c1ac-3a5b-4ae1-a701-fca79955f0a7	B00EBURPLC	203	123	80	0
6	0728VUGV6F2	f945c1ac-3a5b-4ae1-a701-fca79955f0a7	B07VN27QCK	203	203	0	0
7	0728VSD7MCE	f945c1ac-3a5b-4ae1-a701-fca79955f0a7	B07VR8M1XT	203	203	0	0
8	0728VXGXKHP	f945c1ac-3a5b-4ae1-a701-fca79955f0a7	B07VP1YB5L	203	203	0	0
9	0728VSD7JQZ	f945c1ac-3a5b-4ae1-a701-fca79955f0a7	B07VLWH9Y6	203	203	0	0
10	1122VGRNO4V	1134fcd3-ebad-4750-8490-abf840129c62	B081WBBBBZ	203	0	203	0

Based on these results, we can take this further and calculate the proportions. For example for this seller,

mp_sup_key	ASIN	Total Listings	% Active Listings	% Inactive Listings
ceaf28cb-ed08-450f-a87b-596b668d6998	8415579063	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	618789677	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	760735360	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	898708176	137	56.93	43.07
ceaf28cb-ed08-450f-a87b-596b668d6998	131704397	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	B071FJG8RV	137	51.82	48.18
ceaf28cb-ed08-450f-a87b-596b668d6998	931866480	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	140143505	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	1250621569	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	910034737	137	0	100
ceaf28cb-ed08-450f-a87b-596b668d6998	451471814	137	65.69	34.31
ceaf28cb-ed08-450f-a87b-596b668d6998	792279042	137	0	100

This calculation is based on this sql query.

```
SELECT
  mp_sup_key,
  asin,
  COUNT(*) as total_listings,
  ROUND(COUNT(IF(status='Active', 1, NULL))/COUNT(*) * 100, 2) as active_listings,
  ROUND(COUNT(IF(status='Inactive', 1, NULL))/COUNT(*) * 100, 2) as inactive_listings
FROM
  `bigqueryexport-183608.amazon.listings`
WHERE
  --mp_sup_key = 'ce60c2fb-3e7b-49da-8a45-9ce41f439618'
  asin IS NOT NULL
  AND LEFT(open_date, 7) = '2021-01'
GROUP BY
  listing_id,
  mp_sup_key,
  asin
ORDER BY
  total_listings desc
```

The reason I limited my query to 2021 data was to manage data load on google sheets.

This analysis answers two questions:

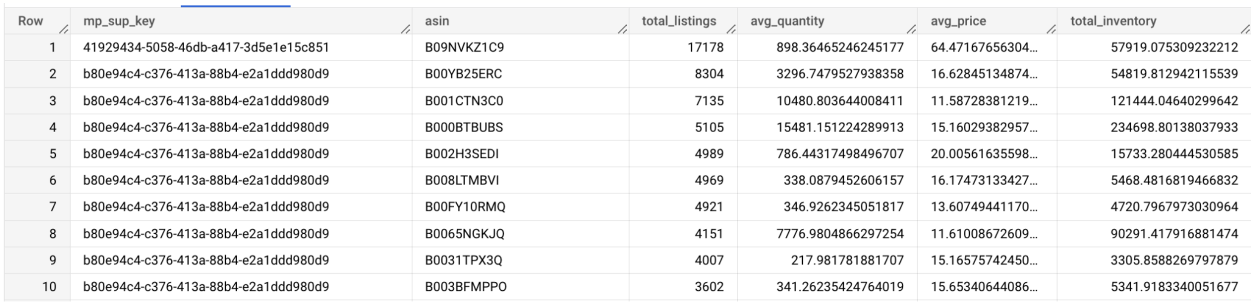
1. How many listings does a seller have on Amazon marketplace?
2. What are the proportions of their active versus inactive listings?

Calculate Inventory value of a seller:

Naive approach: One issue with this approach is that there could be duplicate values for a particular listing_id. But if I group by listing_id, then the analysis is too granular and not very relevant to understand inventory behavior of a seller.

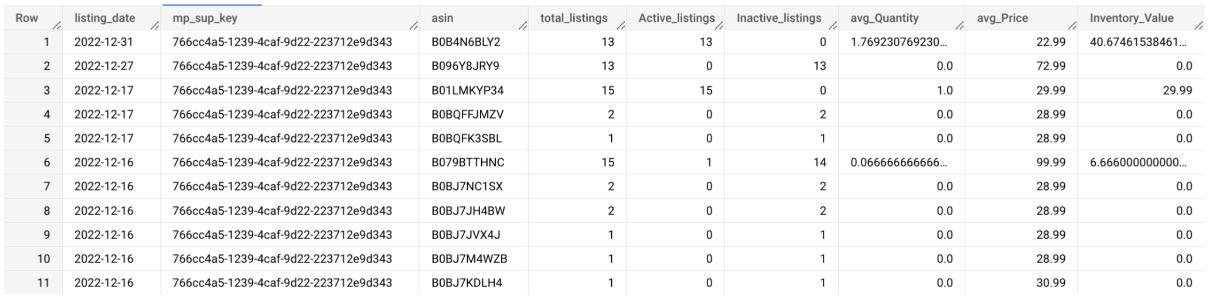
This also includes all the improbable values for price and quantity.

```
SELECT
 mp_sup_key,
 asin,
 COUNT(*) as total_listings,
 AVG(IFNULL(SAFE_CAST(quantity AS FLOAT64), 0.0)) as avg_quantity,
 AVG(IFNULL(SAFE_CAST(price AS FLOAT64), 0.0)) as avg_price,
 AVG(IFNULL(SAFE_CAST(quantity AS FLOAT64), 0.0)) * AVG(IFNULL(SAFE_CAST(price AS
FLOAT64), 0.0)) as total_inventory
FROM
 `bigqueryexport-183608.amazon.listings`
WHERE
 mp_sup_key IS NOT NULL
 AND fulfillment_channel NOT IN ('0','') AND fulfillment_channel IS NOT NULL
 AND status NOT IN ('0','') AND status IS NOT NULL
 AND price <> ''
 AND quantity <> ''
 AND (LEFT(open_date,4)) = '2022'
GROUP BY
 mp_sup_key,
 asin
ORDER BY
 total_listings DESC
 LIMIT 15;
```



Row	mp_sup_key	asin	total_listings	avg_quantity	avg_price	total_inventory
1	41929434-5058-46db-a417-3d5e1e15c851	B09NVKZ1C9	17178	898.36465246245177	64.47167656304...	57919.075309232212
2	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B00YB25ERC	8304	3296.7479527938358	16.62845134874...	54819.812942115539
3	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B001CTN3C0	7135	10480.803644008411	11.58728381219...	121444.04640299642
4	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B000BTBUBS	5105	15481.151224289913	15.16029382957...	234698.80138037933
5	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B002H3SEDI	4989	786.44317498496707	20.00561635598...	15733.280444530585
6	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B008LTMBVI	4969	338.0879452606157	16.17473133427...	5468.4816819466832
7	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B00FY10RMQ	4921	346.9262345051817	13.60749441170...	4720.7967973030964
8	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B0065NGKJQ	4151	7776.9804866297254	11.61008672609...	90291.417916881474
9	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B0031TPX3Q	4007	217.981781881707	15.16575742450...	3305.8588269797879
10	b80e94c4-c376-413a-88b4-e2a1ddd980d9	B003BFMPPO	3602	341.26235424764019	15.65340644086...	5341.9183340051677

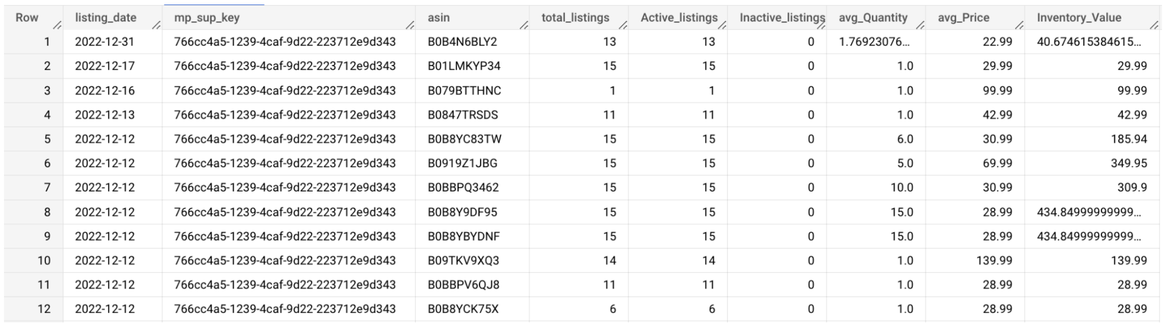
Taking this a step further, we can add the open_date as the listing_date and create a time series data visualization. This is a preview of the data for one seller.



Row	listing_date	mp_sup_key	asin	total_listings	Active_listings	Inactive_listings	avg_Quantity	avg_Price	Inventory_Value
1	2022-12-31	766cc4a5-1239-4caf-9d22-223712e9d343	B0B4N6BLY2	13	13	0	1.769230769230...	22.99	40.67461538461...
2	2022-12-27	766cc4a5-1239-4caf-9d22-223712e9d343	B096Y8JRY9	13	0	13	0.0	72.99	0.0
3	2022-12-17	766cc4a5-1239-4caf-9d22-223712e9d343	B01LMKYP34	15	15	0	1.0	29.99	29.99
4	2022-12-17	766cc4a5-1239-4caf-9d22-223712e9d343	B0BQFFJMZV	2	0	2	0.0	28.99	0.0
5	2022-12-17	766cc4a5-1239-4caf-9d22-223712e9d343	B0BQFK3SBL	1	0	1	0.0	28.99	0.0
6	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B079BTTHNC	15	1	14	0.066666666666...	99.99	6.666000000000...
7	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B0BJ7NC1SX	2	0	2	0.0	28.99	0.0
8	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B0BJ7JH4BW	2	0	2	0.0	28.99	0.0
9	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B0BJ7JVX4J	1	0	1	0.0	28.99	0.0
10	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B0BJ7M4WZB	1	0	1	0.0	28.99	0.0
11	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B0BJ7KDLH4	1	0	1	0.0	30.99	0.0

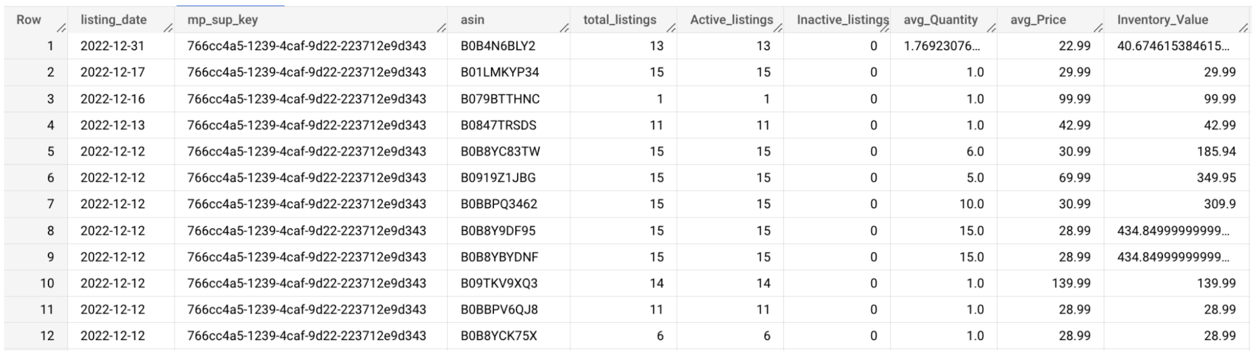
```sql
SELECT
 CAST(LEFT(open_date,10) AS DATE) AS listing_date,
 mp_sup_key,
 asin,
 COUNT(*) as total_listings,
 COUNT(IF(status='Active', 1, NULL)) as Active_listings,
 COUNT(IF(status='Inactive', 1, NULL)) as Inactive_listings,
 AVG(IFNULL(SAFE_CAST(quantity AS INT64), 0.0)) as avg_Quantity,
 AVG(IFNULL(SAFE_CAST(price AS FLOAT64), 0.0)) as avg_Price,
 AVG(IFNULL(SAFE_CAST(price AS FLOAT64), 0.0)) *
 AVG(IFNULL(SAFE_CAST(quantity AS INT64), 0.0)) AS Inventory_Value,
FROM `bigqueryexport-183608.amazon.listings`
WHERE
 mp_sup_key IS NOT NULL
 AND fulfillment_channel NOT IN ('0','') AND fulfillment_channel IS NOT NULL
 AND status NOT IN ('0','') AND status IS NOT NULL
 AND quantity <> ''
 AND price <> ''
 AND asin IS NOT NULL
 AND LEFT(open_date, 4) = '2022'
GROUP BY listing_id, mp_sup_key, asin, open_date
ORDER BY listing_date DESC, Active_listings DESC,total_listings DESC;
```

Based on the results of the above query, my assumption is that only listings with active status have a quantity associated with them.



Row	listing_date	mp_sup_key	asin	total_listings	Active_listings	Inactive_listings	avg_Quantity	avg_Price	Inventory_Value
1	2022-12-31	766cc4a5-1239-4caf-9d22-223712e9d343	B0B4N6BLY2	13	13	0	1.76923076...	22.99	40.674615384615...
2	2022-12-17	766cc4a5-1239-4caf-9d22-223712e9d343	B01LMKYP34	15	15	0	1.0	29.99	29.99
3	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B079BTTHNC	1	1	0	1.0	99.99	99.99
4	2022-12-13	766cc4a5-1239-4caf-9d22-223712e9d343	B0847TRSDS	11	11	0	1.0	42.99	42.99
5	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8YC83TW	15	15	0	6.0	30.99	185.94
6	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0919Z1JBG	15	15	0	5.0	69.99	349.95
7	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0BBPQ3462	15	15	0	10.0	30.99	309.9
8	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8Y9DF95	15	15	0	15.0	28.99	434.84999999999...
9	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8YBYDNF	15	15	0	15.0	28.99	434.84999999999...
10	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B09TKV9XQ3	14	14	0	1.0	139.99	139.99
11	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0BBPV6QJ8	11	11	0	1.0	28.99	28.99
12	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8YCK75X	6	6	0	1.0	28.99	28.99

Filtering all records where price and quantity are not 0, we can see that the total_listings value match active_listings value. So, the analysis can focus only on active_listings in the future.



Row	listing_date	mp_sup_key	asin	total_listings	Active_listings	Inactive_listings	avg_Quantity	avg_Price	Inventory_Value
1	2022-12-31	766cc4a5-1239-4caf-9d22-223712e9d343	B0B4N6BLY2	13	13	0	1.76923076...	22.99	40.674615384615...
2	2022-12-17	766cc4a5-1239-4caf-9d22-223712e9d343	B01LMKYP34	15	15	0	1.0	29.99	29.99
3	2022-12-16	766cc4a5-1239-4caf-9d22-223712e9d343	B079BTTHNC	1	1	0	1.0	99.99	99.99
4	2022-12-13	766cc4a5-1239-4caf-9d22-223712e9d343	B0847TRSDS	11	11	0	1.0	42.99	42.99
5	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8YC83TW	15	15	0	6.0	30.99	185.94
6	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0919Z1JBG	15	15	0	5.0	69.99	349.95
7	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0BBPQ3462	15	15	0	10.0	30.99	309.9
8	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8Y9DF95	15	15	0	15.0	28.99	434.84999999999...
9	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8YBYDNF	15	15	0	15.0	28.99	434.84999999999...
10	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B09TKV9XQ3	14	14	0	1.0	139.99	139.99
11	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0BBPV6QJ8	11	11	0	1.0	28.99	28.99
12	2022-12-12	766cc4a5-1239-4caf-9d22-223712e9d343	B0B8YCK75X	6	6	0	1.0	28.99	28.99

One final issue with this code is the existence of duplicate values. Next step is to find a way to remove them and also validate that there are no duplicates in the results.

Here is a link to the dashboard that visualizes this information.