Since the documentation for these sprints was back-dated as built after discussing with Professor in Sprint 4, it has not been possible to document the most details as not in real time. It showcases the main queries presented and some EDA that in general I remember I had done. A lot of takeaways explained  was after manually checking for one seller (dive deeper) but at this point - I do not remember most of those queries.

## STATEMENTS

EDA on the data:

```
SELECT mp_sup_key, SUM(original_total_amount) AS amount_paid FROM
`bigqueryexport-183608.amazon.statements`
WHERE financial_event_group_start > "2023-01-01" AND original_total_currency= 'USD'
GROUP BY mp_sup_key
ORDER BY amount_paid DESC
```

This is to identify the top sellers for a given time period. Understood that a given financial_event_id exists that basically is a range between a financial start date and end date and this ID basically identifies the start-end period. Filtering on the start date here.

It would be useful to leverage the data to understand the maximum amount due from AMAZON to a particular seller.

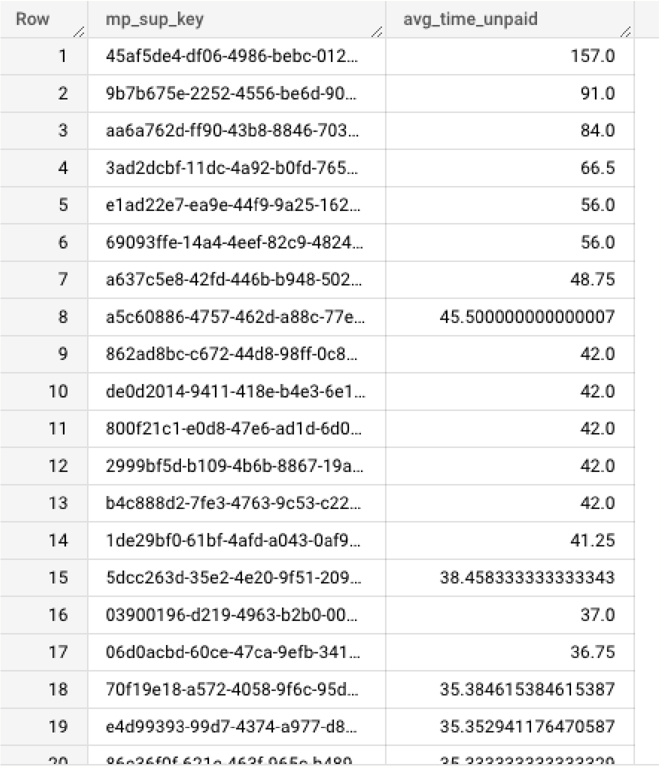
Before writing the following query I did some EDA on the history of statements  a given supplier over time to understand a few attributes:
1. The Processing Status can have two values  - 'open' and 'closed'. If it is closed, it means that a payment was made and was successful. Open means that payment has still not been made from Amazon. Payments can fail too and that you can see in the fund_transfer_status and notes columns.
2. Orginial_total_amount seems to be the amount of money that is being payout. One more parameter is the balance. If sometimes payout is not given can have negative values and then once original_total_amount is given in the next transaction, balance should get updated - Gregor has mentioned. But not been able to verify yet

```
SELECT mp_sup_key, COUNT(*) AS number_of_open_transactions,
SUM(original_total_amount) AS amount_due_from_amazon FROM
```

```
`bigqueryexport-183608.amazon.statements`
WHERE financial_event_group_start > "2023-01-01" AND original_total_currency= 'USD'AND
processing_status= 'Open'
GROUP BY mp_sup_key
ORDER BY amount_due_from_amazon DESC
```

Now understand the maximum time they are unpaid- payability can leverage this data to offer them promotions and loans because they are not getting paid by amazon. (Also ask Team why are they not getting paid for so long)

```
SELECT mp_sup_key, AVG(DATE_DIFF(financial_event_group_end,
financial_event_group_start, DAY)) AS avg_time_unpaid FROM
`bigqueryexport-183608.amazon.statements`
WHERE financial_event_group_start > "2022-01-01" AND original_total_currency= 'USD'AND
processing_status= 'Closed'
GROUP BY mp_sup_key
ORDER BY avg_time_unpaid DESC;
```

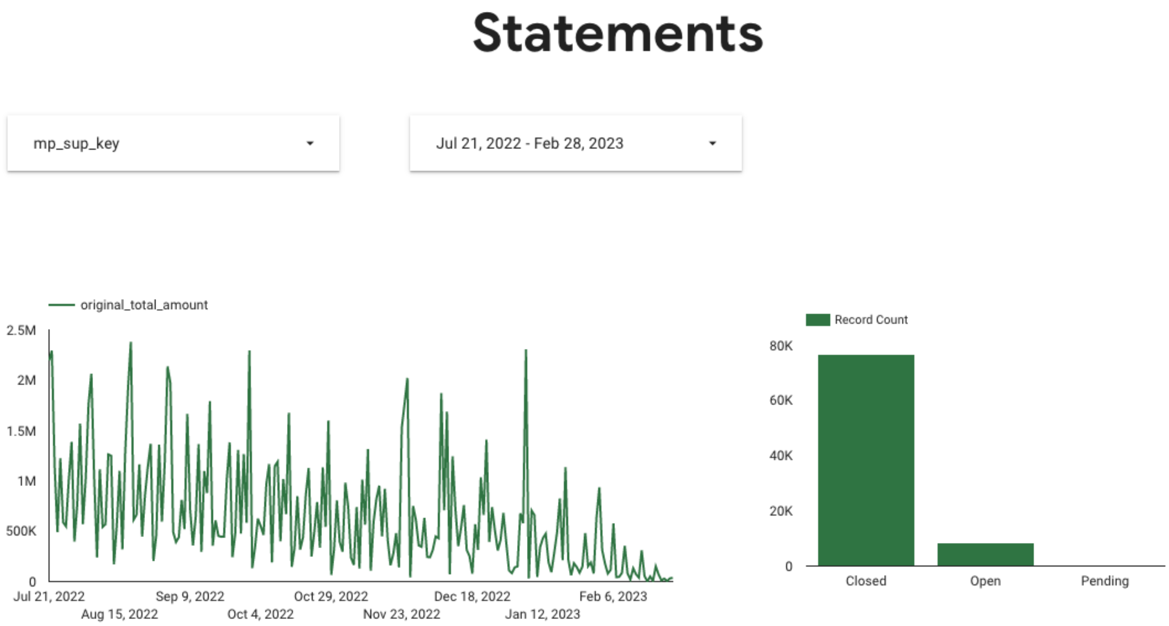


| Row | mp_sup_key | avg_time_unpaid |
|---|---|---|
| 1 | 45af5de4-df06-4986-bebc-012... | 157.0 |
| 2 | 9b7b675e-2252-4556-be6d-90... | 91.0 |
| 3 | aa6a762d-ff90-43b8-8846-703... | 84.0 |
| 4 | 3ad2dcbf-11dc-4a92-b0fd-765... | 66.5 |
| 5 | e1ad22e7-ea9e-44f9-9a25-162... | 56.0 |
| 6 | 69093ffe-14a4-4eef-82c9-4824... | 56.0 |
| 7 | a637c5e8-42fd-446b-b948-502... | 48.75 |
| 8 | a5c60886-4757-462d-a88c-77e... | 45.500000000000007 |
| 9 | 862ad8bc-c672-44d8-98ff-0c8... | 42.0 |
| 10 | de0d2014-9411-418e-b4e3-6e1... | 42.0 |
| 11 | 800f21c1-e0d8-47e6-ad1d-6d0... | 42.0 |
| 12 | 2999bf5d-b109-4b6b-8867-19a... | 42.0 |
| 13 | b4c888d2-7fe3-4763-9c53-c22... | 42.0 |
| 14 | 1de29bf0-61bf-4afd-a043-0af9... | 41.25 |
| 15 | 5dcc263d-35e2-4e20-9f51-209... | 38.458333333333343 |
| 16 | 03900196-d219-4963-b2b0-00... | 37.0 |
| 17 | 06d0acbd-60ce-47ca-9efb-341... | 36.75 |
| 18 | 70f19e18-a572-4058-9f6c-95d... | 35.384615384615387 |
| 19 | e4d99393-99d7-4374-a977-d8... | 35.352941176470587 |
| 20 | 86e36f0f-621e-463f-965e-b489... | 35.333333333333329 |

The output was very weird to know. Many sellers had a lot of days unpaid. WHY

Comments in general by Gregor later -

1. A lot of sellers might have partnered with Payability and left. So we keep on hitting the Amazon API but we have stopped receiving data for them. That is why some have a lot of days - they are outliers and we should not care cause they are of no use to us now.
2. Keep a check if a seller exists in the recent months and maybe check data for them only



Statements

mp_sup_key

Jul 21, 2022 - Feb 28, 2023

This dashboard was made to see a time-series distribution of payouts to sellers. You can also compare how many closed and open transactions and flag the customers who have more Open >>>> Close. Their account should be in RISK.

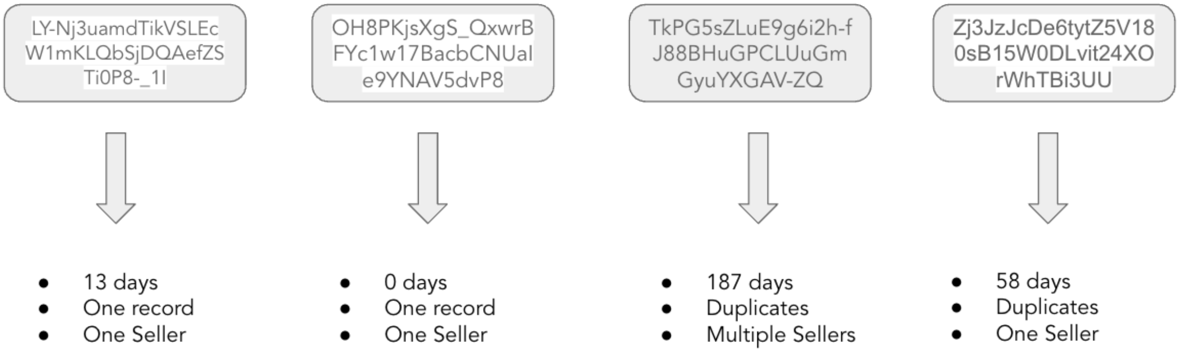
Comments about expectations by Gregor later -

1. Build a time-series chart to see if the payments are being processed timely and the trend followed.
2. Compare payouts of one seller with all sellers in a given financial event

It is all about the financial event so let us verify if the financial event is usually 15 days. ( It was mentioned by Payability team that payouts happen from Amazon to a seller every 15 days. Time to check if it is really happening)

```
SELECT financial_event_group_id , financial_event_group_start, financial_event_group_end,
DATE_DIFF(financial_event_group_end , financial_event_group_start, DAY ) AS
NO_OF_DAYS_IN_AN_EVENT
FROM `bigqueryexport-183608.amazon.statements`
ORDER BY financial_event_group_end DESC ;
```

The days are not at all constant. Varying from 5,19, 400, 100 etc…



LY-Nj3uamdTikVSLEc W1mKLQbSjDQAefZS Ti0P8-_1I

- 13 days
- One record
- One Seller

OH8PKjsXgS_QxwrB FYc1w17BacbCNUaI e9YNAV5dvP8

- 0 days
- One record
- One Seller

TkPG5sZLuE9g6i2h-f J88BHuGPCLUuGm GyuYXGAV-ZQ

- 187 days
- Duplicates
- Multiple Sellers

Zj3JzJcDe6tytZ5V18 0sB15W0DLvit24XO rWhTBi3UU

- 58 days
- Duplicates
- One Seller

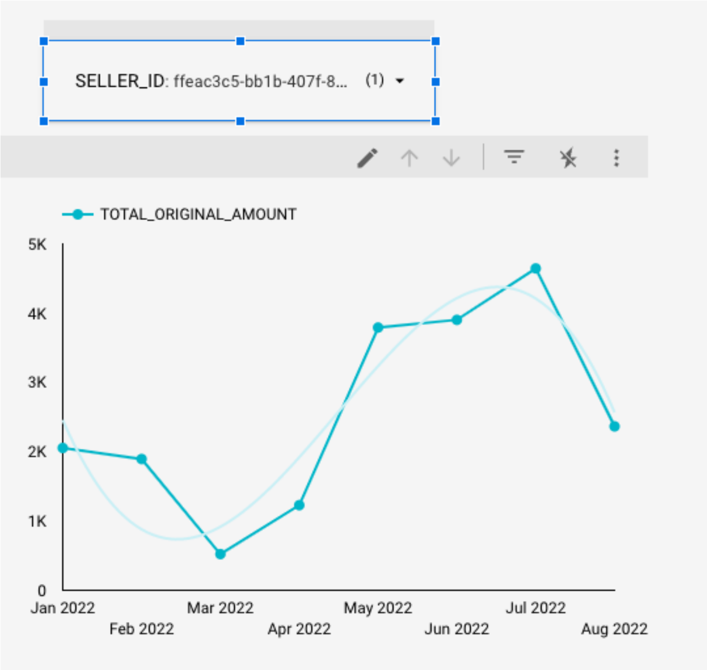
I had ran custom queries for multiple event ids to see how days and records are varying in these time periods. After multiple manual analysis and talks with Gregor had these key takeaways:

1. *original_total_amount - is the statement end balance. If we download it few times day after day the change should reflect daily sales*
2. *As long as the statement is 'OPEN' the financial_event_group_end = download date.*
3. *Plot with respect to end date*

*MAIN TAKEAWAY - Payability does not have this data from Amazon. They are hitting the API nearly every other day so they have many duplicates. So you will have to group by the statemnt_id and the seller. Also extract the end date and plot wrt to build the time series chart*

```
SELECT financial_event_group_id AS STATEMENT_ID , mp_sup_key AS SELLER_ID ,
SUM(original_total_amount) AS TOTAL_ORIGINAL_AMOUNT ,
MAX(financial_event_group_end) AS END_DATE
FROM `bigqueryexport-183608.amazon.statements`
WHERE EXTRACT(YEAR FROM financial_event_group_start) IN (2021 ,2022, 2023) AND
original_total_currency= 'USD'
GROUP BY financial_event_group_id , mp_sup_key
ORDER BY END_DATE ;
```

*While plotting the dashboard understand how to group by the Year-Month in Google Looker Studio so getting data for every day would be the best .*



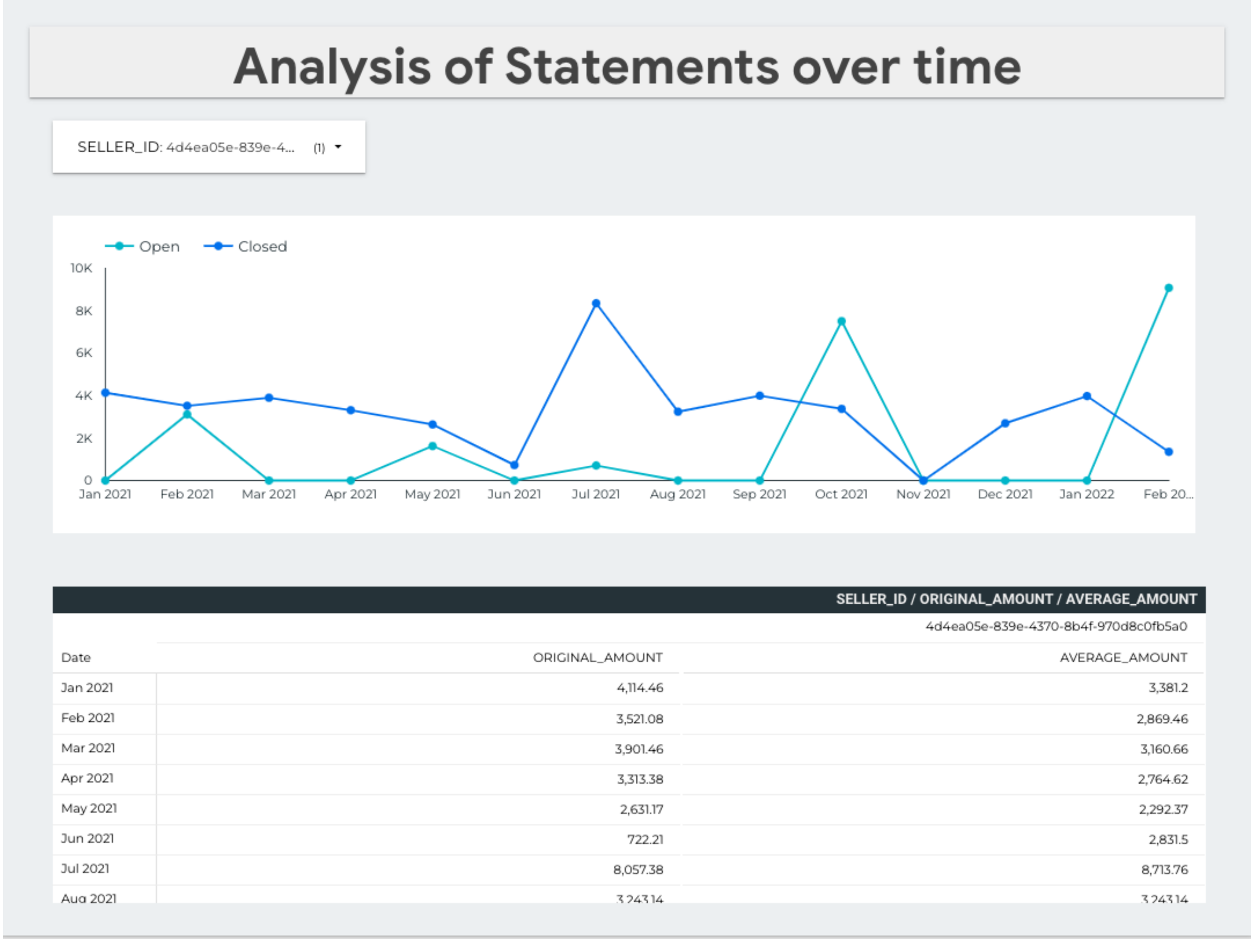
*But you also want to compare a sellers payout with all the sellers. Maintain a global average that just group's by the date and not the seller unlike the previous query*

```
SELECT C.financial_event_group_id AS STATEMENT_ID, C.mp_sup_key AS SELLER_ID,
C.original_total_amount AS ORIGINAL_AMOUNT, C.processing_status AS
PROCESSING_STATUS,  D.END_DATE AS END_DATE , D.AVERAGE_AMOUNT
```

```
FROM `bigqueryexport-183608.amazon.statements` C
INNER JOIN
(SELECT END_DATE AS END_DATE, AVG(ORIGINAL_AMOUNT) AS AVERAGE_AMOUNT
FROM
(SELECT A.financial_event_group_id AS STATEMENT_ID, A.mp_sup_key AS SELLER_ID,
A.original_total_amount AS ORIGINAL_AMOUNT, A.processing_status AS
PROCESSING_STATUS,  B.END_DATE AS END_DATE
FROM `bigqueryexport-183608.amazon.statements` A
INNER JOIN
(SELECT financial_event_group_id AS STATEMENT_ID , mp_sup_key AS SELLER_ID,
MAX(financial_event_group_end) AS END_DATE
FROM `bigqueryexport-183608.amazon.statements`
WHERE EXTRACT(YEAR FROM financial_event_group_start) IN (2021 ,2022, 2023) AND
original_total_currency= 'USD'
GROUP BY financial_event_group_id , mp_sup_key
ORDER BY financial_event_group_id) B
ON A.financial_event_group_id= B.STATEMENT_ID AND A.mp_sup_key= B.SELLER_ID
ORDER BY END_DATE)
GROUP BY END_DATE) D
ON C.financial_event_group_end= D.END_DATE
ORDER BY STATEMENT_ID ;
```

Took multiple inner joins with the same table. First, I also wanted other columns like Processing status , date which are not a part of aggregation in the basic query (only statement_id and seller). So first fetch those other columns as well. Processing status can be utilized to add granularity in the visualization and also try visualizing if what Gregor said is actually true-  does open and close get updated after every cycle?
Once you do that, you need to join once again because you will have made a sub query to find the global average of the payouts of all sellers on one day (end_Date). So add that too.

After all this pivot the table where you group by Year-Month on Looker and have two columns - normal sales and total sales.



## Analysis of Statements over time

SELLER_ID: 4d4ea05e-839e-4... (1) ▾

| | | SELLER_ID / ORIGINAL_AMOUNT / AVERAGE_AMOUNT |
|---|---|---|
| | | 4d4ea05e-839e-4370-8b4f-970d8c0fb5a0 |
| Date | ORIGINAL_AMOUNT | AVERAGE_AMOUNT |
| Jan 2021 | 4,114.46 | 3,381.2 |
| Feb 2021 | 3,521.08 | 2,869.46 |
| Mar 2021 | 3,901.46 | 3,160.66 |
| Apr 2021 | 3,313.38 | 2,764.62 |
| May 2021 | 2,631.17 | 2,292.37 |
| Jun 2021 | 722.21 | 2,831.5 |
| Jul 2021 | 8,057.38 | 8,713.76 |
| Aug 2021 | 3,243.14 | 3,243.14 |

*Inflated values due to duplicates so validate with the team.*

Nayan and Amy had no comments about the validity during the presentation as they do not deal with this data.

## CUSTOMER ORDER METRICS:

Gives us information about the orders delivered from a seller to a consumer.

```
SELECT DISTINCT SALES_CHANNEL FROM
`bigqueryexport-183608.amazon.customer_order_metrics` ;
```

Ouput contains - Amazon.com.br - brazil , Amazon.ca - canada , Amazon.com - general , Non-Amazon ,SI CA Prod Marketplace- dontknow , Amazon.com.mx - mexico (All the understandings made from Google and confirmed later)

SELECT DISTINCT order_status FROM
`bigqueryexport-183608.amazon.customer_order_metrics` ;

Output - 19 categories in shipped, pending and complete : DOUBT : Confirm which one to consider

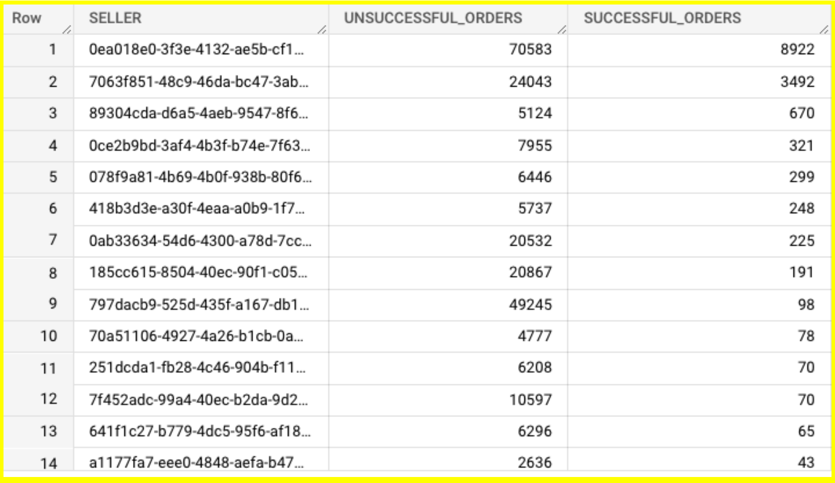
Focused on US only so planned on filtering the Sales_channel. Also, wanted to understand from Payability team which categories do we consider in sales because Shipped has many subcategories like shipped- delivered to the buyer, shipped- in transit, shipped- seller returned it. So for the other categories do we actually calculate the sales - ASK

Is_fba means that it is fulfilled by Amazon.

Trying to see the proportion of unsuccessful to successful orders. WHY?
My idea was to find this proportion and join with the Account_Status table to see if there is a correlation between this proportion and the Status.

(COUNT(IF() was not working had to find a fix for it)

```
SELECT MP_SUP_KEY AS SELLER,
COUNTIF(ORDER_STATUS IN ('Shipped - Rejected by Buyer', 'Shipped - Lost in Transit',
'Shipped - Returning to Seller', 'Shipped - Damaged', 'Cancelled')) AS
UNSUCCESSFUL_ORDERS,
COUNTIF(ORDER_STATUS IN ('Complete', 'Shipped - Delivered to Buyer')) as
SUCCESSFUL_ORDERS
FROM `bigqueryexport-183608.amazon.customer_order_metrics`
WHERE EXTRACT(YEAR FROM purchase_date) IN (2021 ,2022, 2023) AND SALES_CHANNEL =
'Amazon.com'
GROUP BY MP_SUP_KEY
order by SUCCESSFUL_ORDERS DESC;
```



| Row | SELLER | UNSUCCESSFUL_ORDERS | SUCCESSFUL_ORDERS |
|---|---|---|---|
| 1 | 0ea018e0-3f3e-4132-ae5b-cf1... | 70583 | 8922 |
| 2 | 7063f851-48c9-46da-bc47-3ab... | 24043 | 3492 |
| 3 | 89304cda-d6a5-4aeb-9547-8f6... | 5124 | 670 |
| 4 | 0ce2b9bd-3af4-4b3f-b74e-7f63... | 7955 | 321 |
| 5 | 078f9a81-4b69-4b0f-938b-80f6... | 6446 | 299 |
| 6 | 418b3d3e-a30f-4eaa-a0b9-1f7... | 5737 | 248 |
| 7 | 0ab33634-54d6-4300-a78d-7cc... | 20532 | 225 |
| 8 | 185cc615-8504-40ec-90f1-c05... | 20867 | 191 |
| 9 | 797dacb9-525d-435f-a167-db1... | 49245 | 98 |
| 10 | 70a51106-4927-4a26-b1cb-0a... | 4777 | 78 |
| 11 | 251dcda1-fb28-4c46-904b-f11... | 6208 | 70 |
| 12 | 7f452adc-99a4-40ec-b2da-9d2... | 10597 | 70 |
| 13 | 641f1c27-b779-4dc5-95f6-af18... | 6296 | 65 |
| 14 | a1177fa7-eee0-4848-aefa-b47... | 2636 | 43 |

As you can see above - 1. Looks like many duplicates and that the orders are not seeing to be updated. Like Transit - after that what happens ? New ORDER_ID assigned? No idea about the mechanism employed by Amazon.

Sales of individual seller with all sellers :

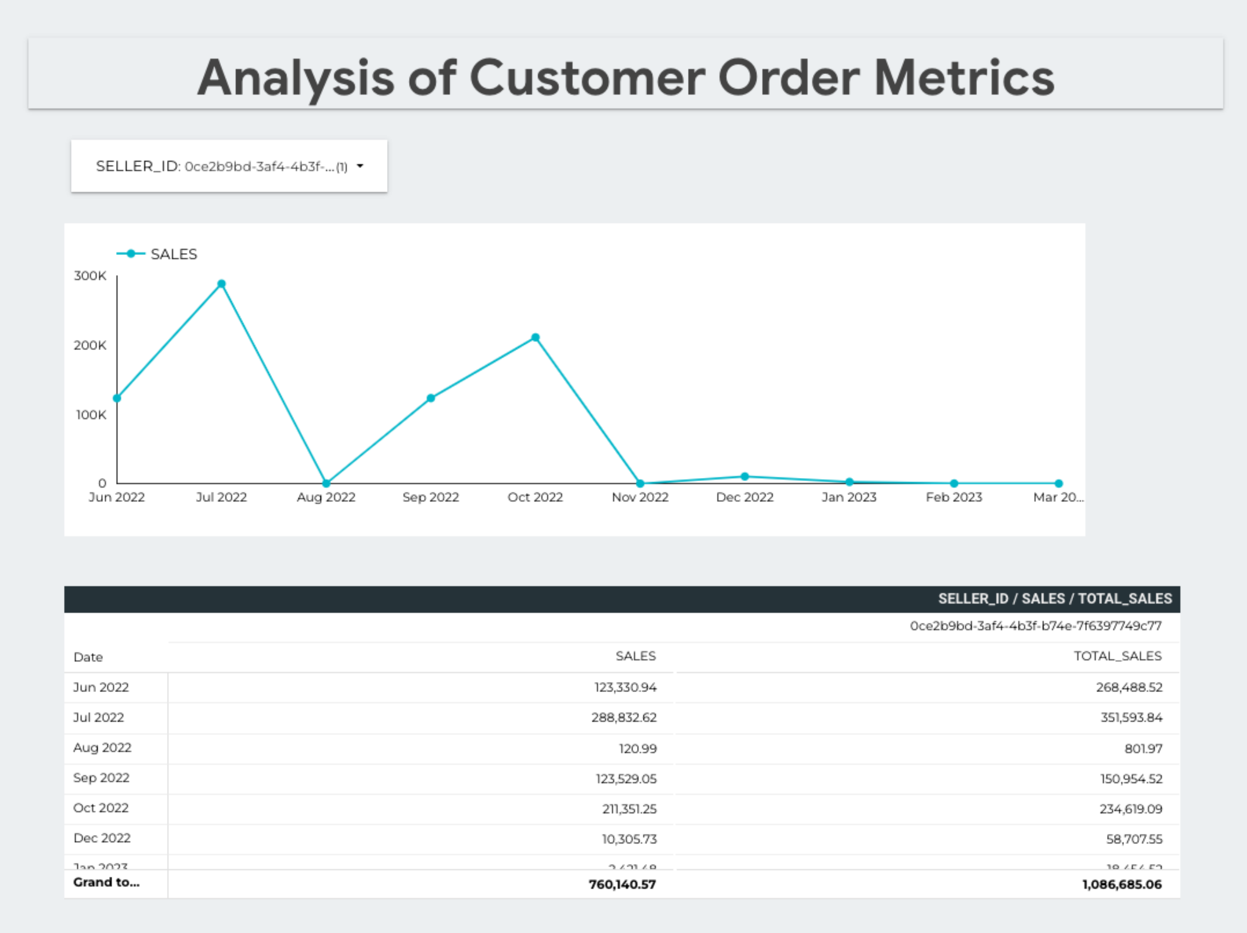
Tried the group by's but was not able to get the performance of total. Wrong values and duplicates were not going.
One of the failed outputs from queries are

```
SELECT MP_SUP_KEY AS SELLER_ID , DATE_PURCHASED,  SUM(item_price * quantity) as
SALES
FROM
(SELECT A.MP_SUP_KEY, A.item_price, A.quantity , B.DATE_PURCHASED,
B.TOTAL_SALES
FROM `bigqueryexport-183608.amazon.customer_order_metrics` A
INNER JOIN
(SELECT purchase_day as DATE_PURCHASED,  SUM(item_price * quantity) AS
TOTAL_SALES
FROM `bigqueryexport-183608.amazon.customer_order_metrics`
WHERE EXTRACT(YEAR FROM purchase_date) IN (2021 ,2022, 2023) AND
SALES_CHANNEL = 'Amazon.com' AND ORDER_STATUS IN ('Complete', 'Shipped -
Delivered to Buyer')
GROUP BY purchase_day
ORDER BY purchase_day) B
ON A.purchase_day = B.DATE_PURCHASED) B
GROUP BY MP_SUP_KEY, DATE_PURCHASED
ORDER BY SELLER_ID
```

Read up about PARTITION BY in BigQuery and chose to use it for easier querying

```
SELECT MP_SUP_KEY AS SELLER_ID ,
PURCHASE_DAY AS DATE_OF_PURCHASE,
SUM(ITEM_PRICE * QUANTITY) OVER (PARTITION BY PURCHASE_DAY,MP_SUP_KEY ) AS SALES,
SUM(ITEM_PRICE * QUANTITY) OVER (PARTITION BY PURCHASE_DAY) AS TOTAL_SALES
FROM `bigqueryexport-183608.amazon.customer_order_metrics`
WHERE EXTRACT(YEAR FROM purchase_date) IN (2021 ,2022, 2023) AND SALES_CHANNEL =
'Amazon.com' AND ORDER_STATUS IN ('Complete', 'Shipped - Delivered to Buyer')
ORDER BY DATE_OF_PURCHASE DESC;
```

Assumption used was that orders successful are the ones with order status filter added in query as other categories did not make sense to include.



Analysis of Customer Order Metrics

SELLER_ID: 0ce2b9bd-3af4-4b3f-...(1) ▾

| | SELLER_ID / SALES / TOTAL_SALES | |
|---|---|---|
| | | 0ce2b9bd-3af4-4b3f-b74e-7f6397749c77 |
| Date | SALES | TOTAL_SALES |
| Jun 2022 | 123,330.94 | 268,488.52 |
| Jul 2022 | 288,832.62 | 351,593.84 |
| Aug 2022 | 120.99 | 801.97 |
| Sep 2022 | 123,529.05 | 150,954.52 |
| Oct 2022 | 211,351.25 | 234,619.09 |
| Dec 2022 | 10,305.73 | 58,707.55 |
| Jan 2023 | 2,421.68 | 18,454.52 |
| **Grand to...** | **760,140.57** | **1,086,685.06** |

A lot of inflated values. I could see way too many duplicates. Order ids getting repeated. And everytime getting added. Need to clean the data more.

Comments discussed during the Presentation:

1. Not sure if the order gets updated. Most probably a new order must be made and no need to care about the old order.
2. Data Validation needs to be done for the sellers. Team has Amazon access for few sellers and it will be shared with us to understand and resolve all the doubts we have about what to filter on and what not. So identify a few common sellers and reach out to get the Amazon dashboard
3. Build a collaborative dashboard with pages.