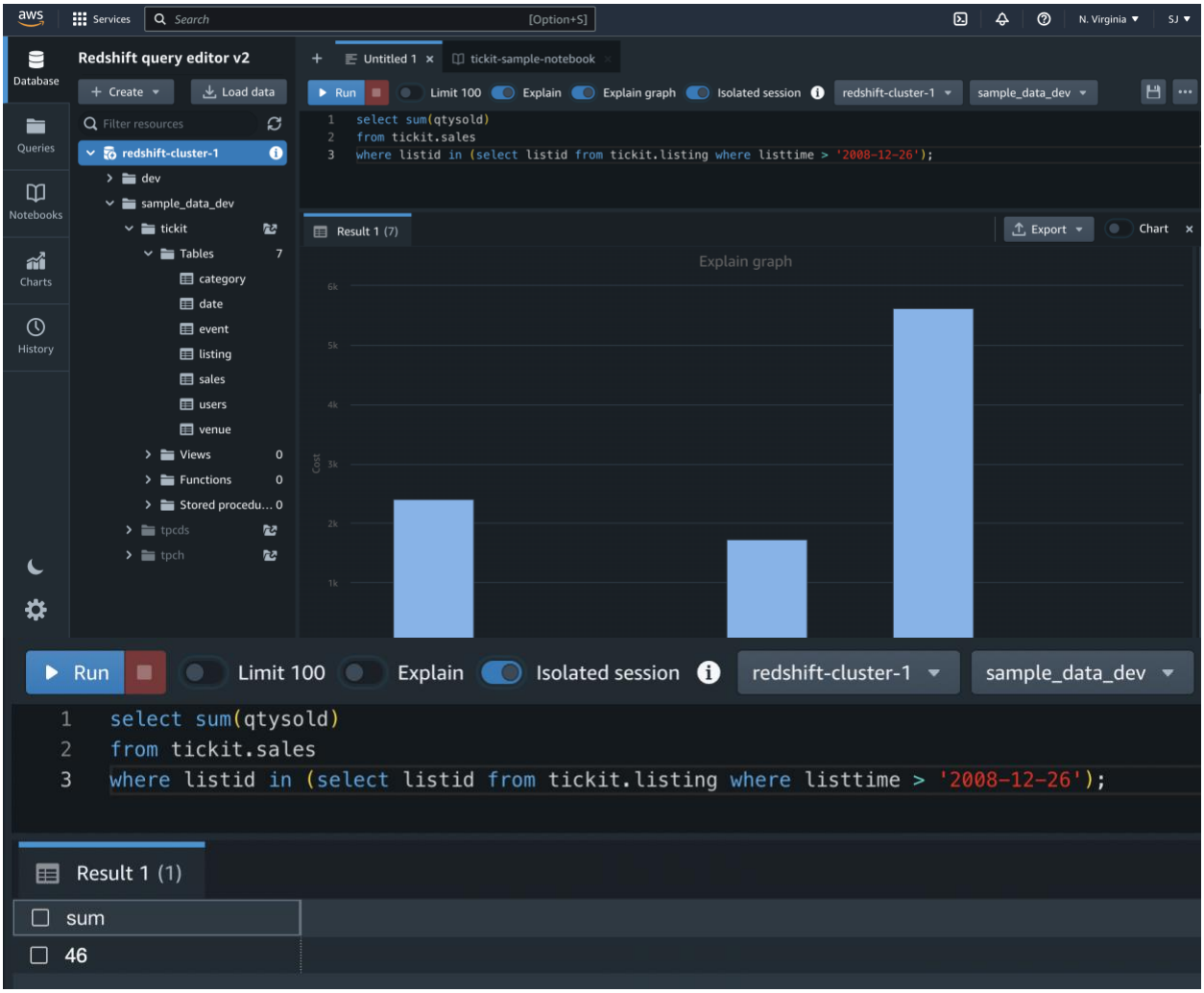
Cloud Data Warehousing

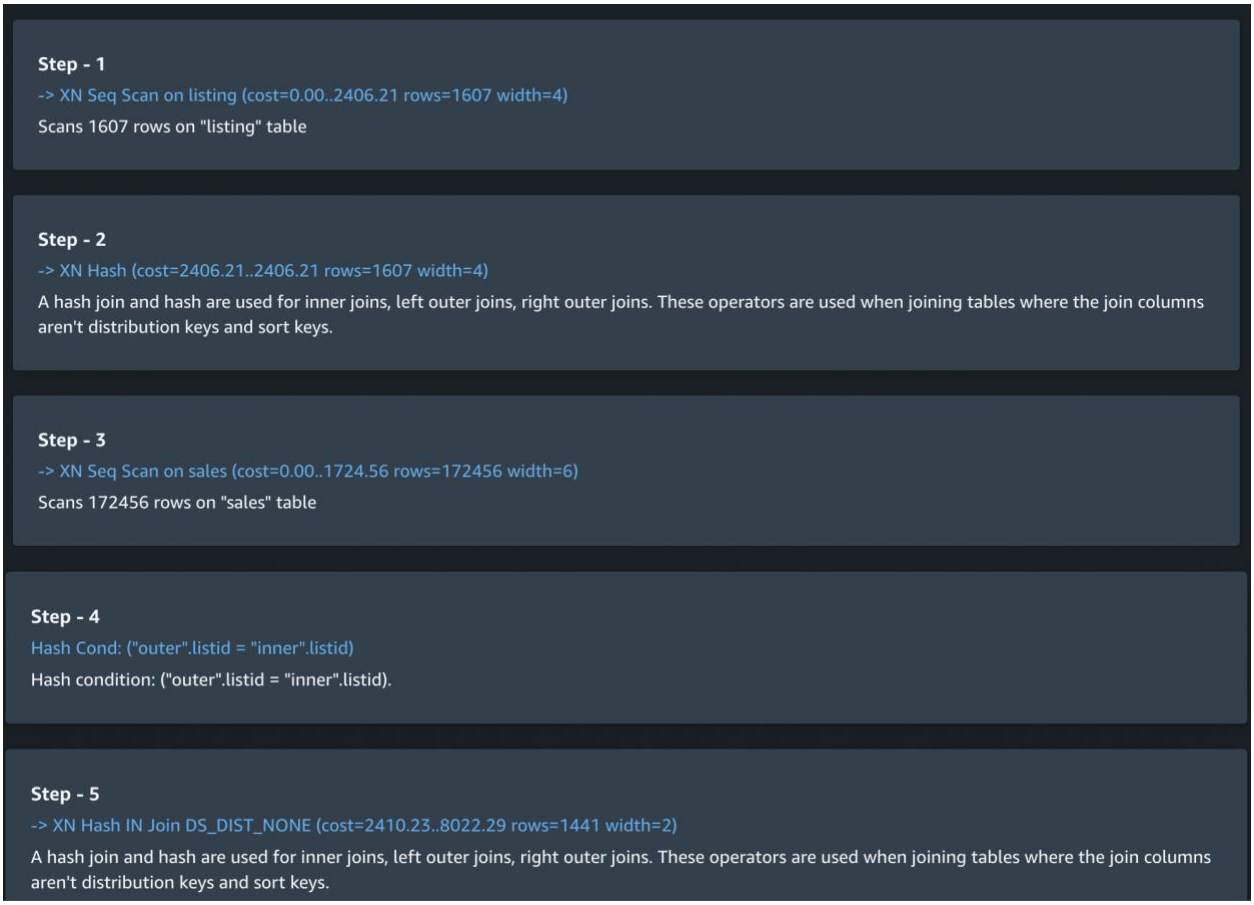
Part 1

Create an AWS Redshift Cluster with the tickitdb sample database.
Compare the costs of the following two queries

Query 1

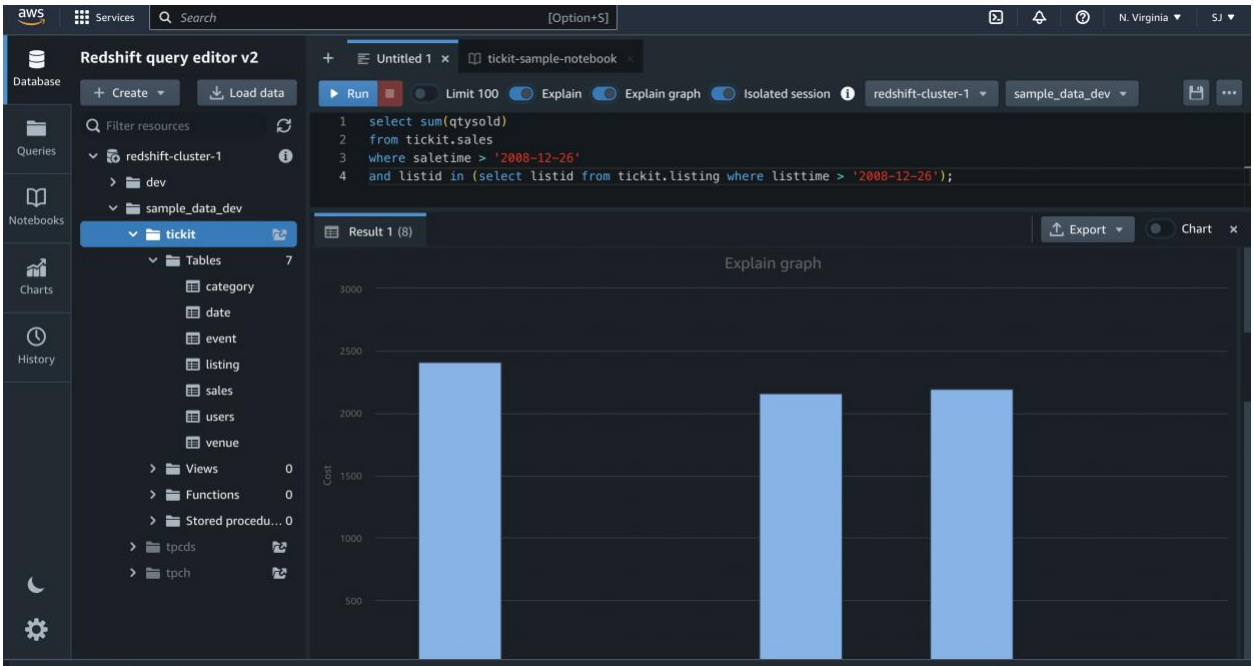
select sum(qtysold)
from sales
where listid in (select listid from listing where listtime > '2008-12-26');

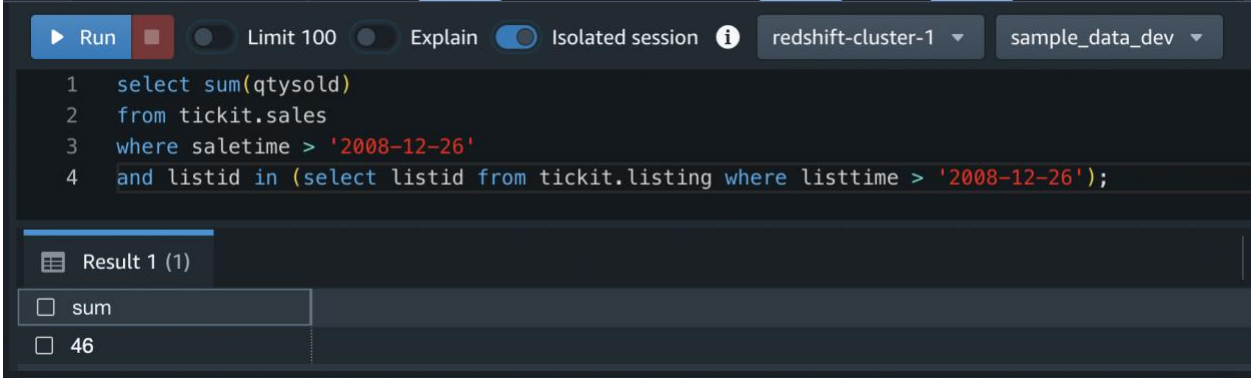




Query 2

select sum(qtysold)
from sales
where saletime > '2008-12-26'
and listid in (select listid from listing where listtime > '2008-12-26');





Step - 1

-> XN Seq Scan on listing (cost=0.00..2406.21 rows=1607 width=4)

Scans 1607 rows on "listing" table

Step - 2

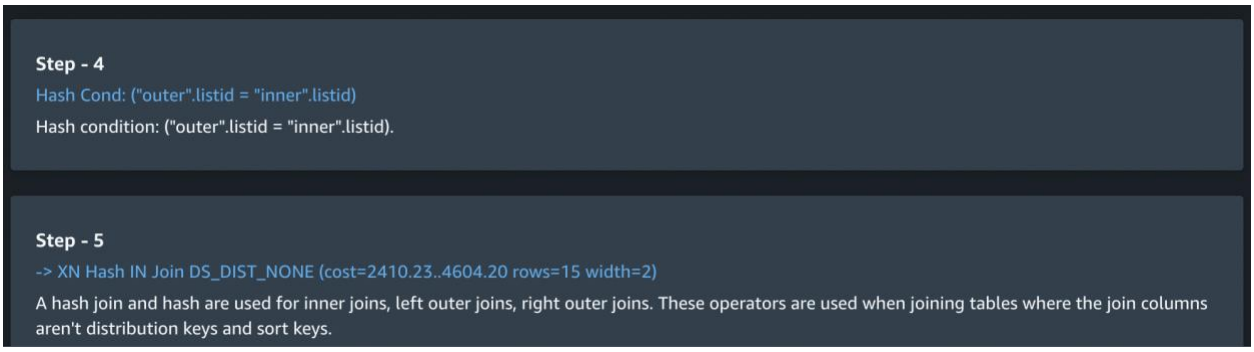
-> XN Hash (cost=2406.21..2406.21 rows=1607 width=4)

A hash join and hash are used for inner joins, left outer joins, right outer joins. These operators are used when joining tables where the join columns aren't distribution keys and sort keys.

Step - 3

-> XN Seq Scan on sales (cost=0.00..2155.70 rows=1794 width=6)

Scans 1794 rows on "sales" table



1st Query is giving us the total quantity of items sold which were listed after 2008-12-26
2nd Query is giving us the total quantity of items which are listed and sold after 2008-12-26

We will get the same output with both the queries because items will be sold only after listing them first, so all the items listed after 2008-12-26 must be sold after 2008-12-26.

Total real cost of 1st query: 2406 + 1724 + 5612 = 9742
Total real cost of 2nd query: 2406 + 2155 + 2194 = 6755

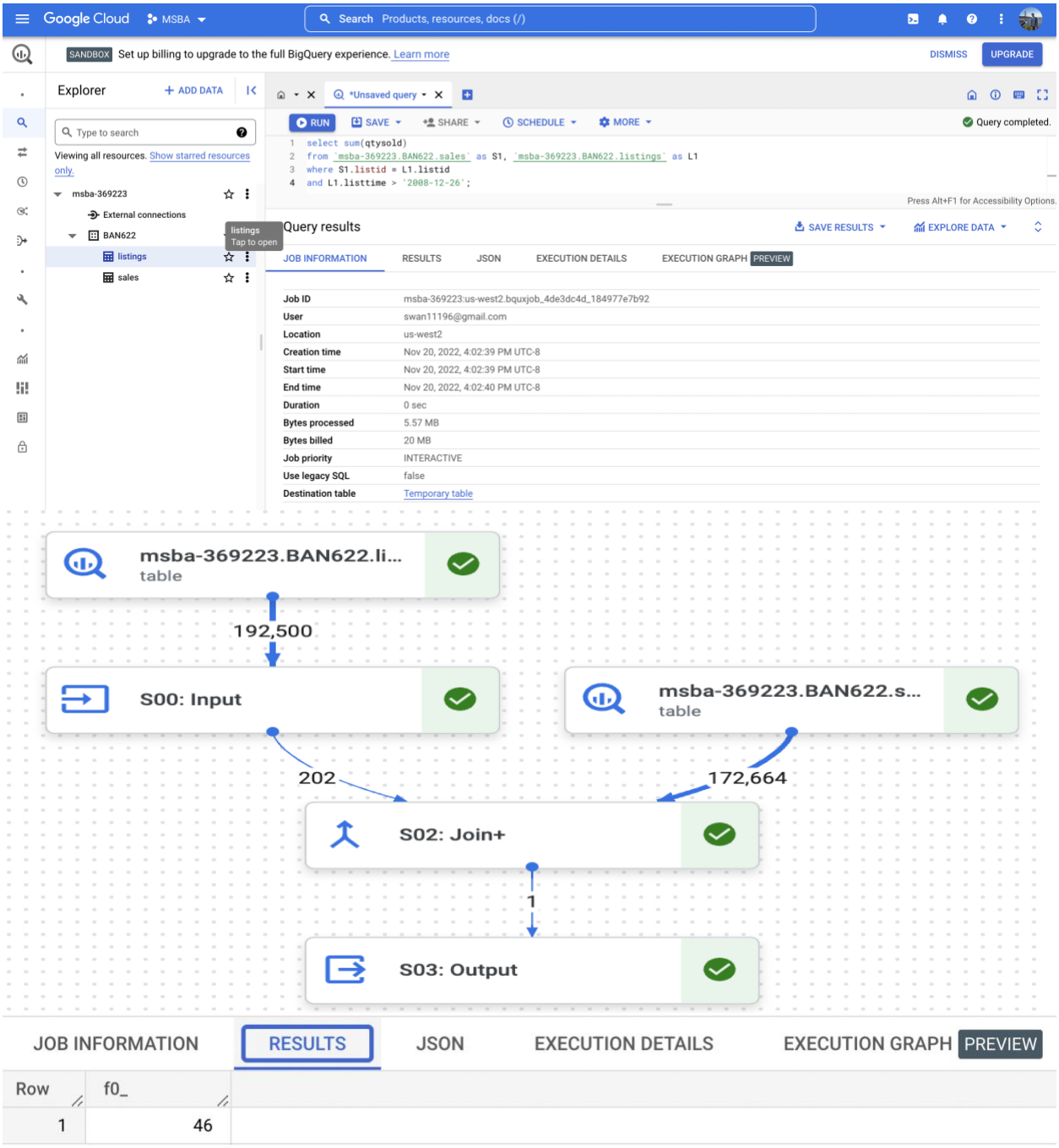
So even if both the queries will give the same output 2nd query is more efficient because as we can see from the steps explained in the above screenshots for 2nd query, we are scanning only 1794 rows of the sales table as compared to scanning 172465 rows of sales table in case of 1st query, and these filtered rows from sales table are compared with the output of query on listing table instead of comparing all the rows from sales table as done in 1st query.

Part 2

Create a BigQuery project and dataset with listing.csv and sales.csv tables
Compare the costs of the following two queries

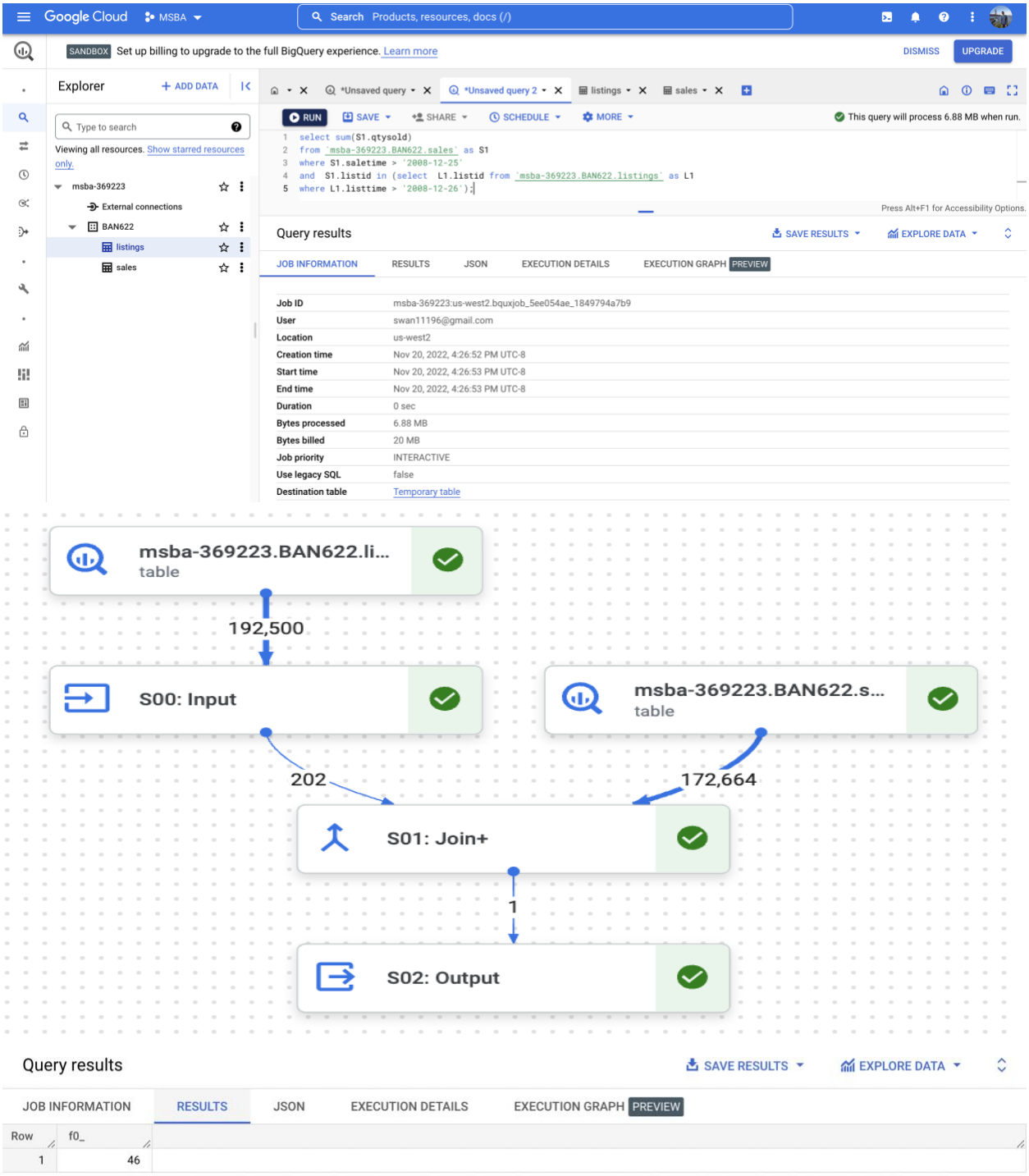
Query 1

```
select sum(qtysold)
from `msba-369223.BAN622.sales` as S1, `msba-369223.BAN622.listings` as L1
where S1.listid = L1.listid
and L1.listtime > '2008-12-26';
```



Query 2

select sum(qtysold)
from `msba-369223.BAN622.sales` as S1
where S1.saletime > '2008-12-25'
and S1.listid in (select listid from `msba-369223.BAN622.listings` as L1
where L1.listtime > '2008-12-26');



1st Query is giving us the total quantity of items sold which were listed after 2008-12-26
2nd Query is giving us the total quantity of items which are sold after 2008-12-25 and listed after 2008-12-26

As mentioned in part one we will get the same output with both the queries because items will be sold only after listing them first, so all the items listed after 2008-12-26 must be sold after 2008-12-25.

cost of 1st query: 5.57MB bytes processed
cost of 2nd query: 6.88MB bytes processed

So even if both the queries will give the same output 1st query is more efficient because we have only one filter in place and even with an inner join operation, we are processing less number bytes as compared to 2nd query.

Price estimate for both queries:

name	quantity	region	service_id	sku	product_description	unit_price, USD	total_price, USD
Analysis (us-west2)	5.31197E-06	us-west2	24E6-581D-38E5	6DFE-432F-94FB	CP-BIGQUERY-GENERAL	0	0
Analysis (us-west2)	6.56128E-06	us-west2	24E6-581D-38E5	6DFE-432F-94FB	CP-BIGQUERY-GENERAL	0	0
						Total Price:	0
						* Sustained use discount (SUD) is not included. You may need to apply discounts separately for each SKU	
Prices are in US dollars, effective date is 2022-11-21T06:40:25.531Z.							
The estimated fees provided by Google Cloud Pricing Calculator are for discussion purposes only and are not binding on either you or Google. Your actual fees may be higher or lower than the estimate.							
Url to the estimate: https://cloud.google.com/products/calculator/#id=010de95e-18a7-44a7-8f63-154e614a3584							