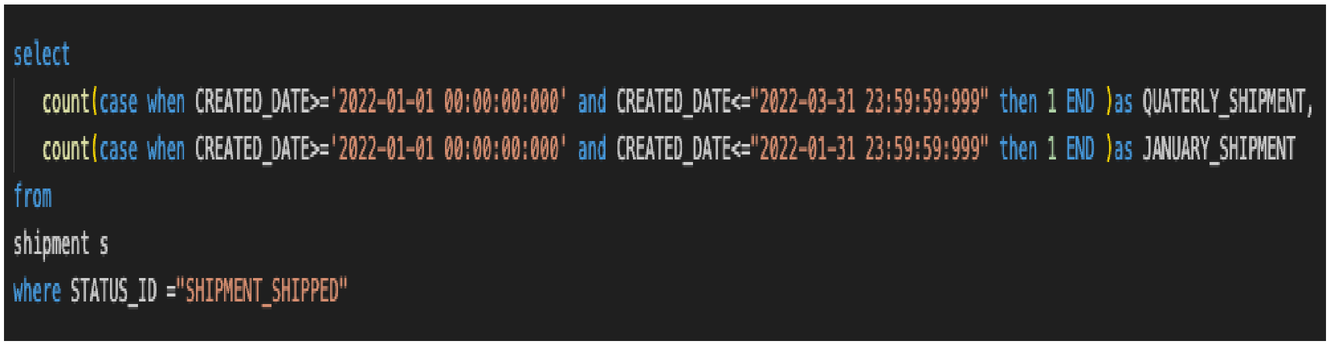
Assignment 1

1. **Total number of shipments in January 2022 first quarter:**

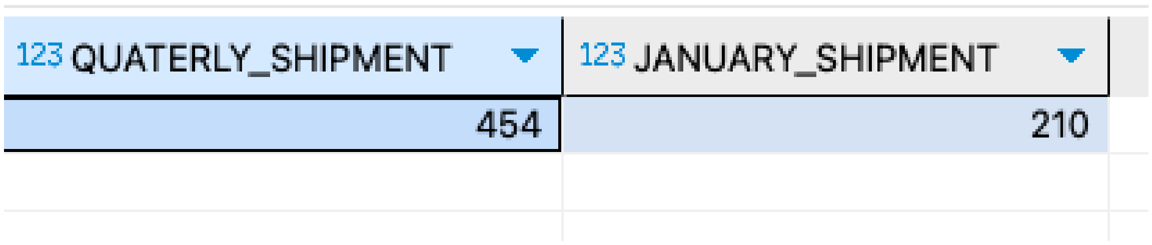
    - Determine the total count of shipments made during the first quarter of 2022, specifically in the month of January.

Solution



```
select
    count(case when CREATED_DATE>='2022-01-01 00:00:00:000' and CREATED_DATE<="2022-03-31 23:59:59:999" then 1 END )as QUATERLY_SHIPMENT,
    count(case when CREATED_DATE>='2022-01-01 00:00:00:000' and CREATED_DATE<="2022-01-31 23:59:59:999" then 1 END )as JANUARY_SHIPMENT
from
shipment s
where STATUS_ID ="SHIPMENT_SHIPPED"
```

Output

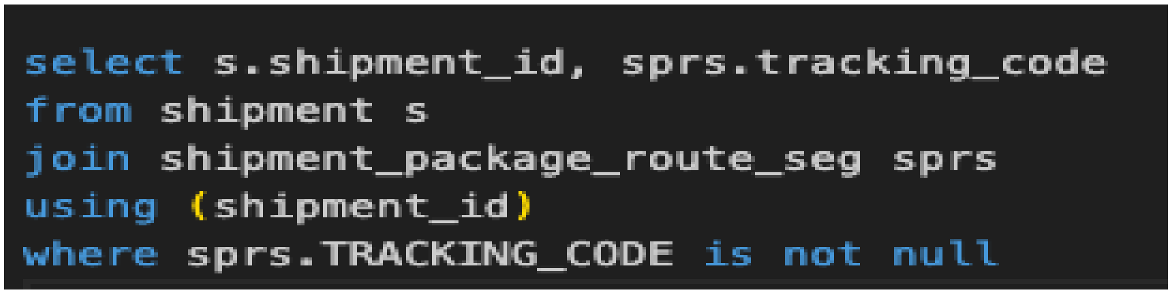


123 QUATERLY_SHIPMENT	123 JANUARY_SHIPMENT
454	210

2. **Shipment by Tracking number:**

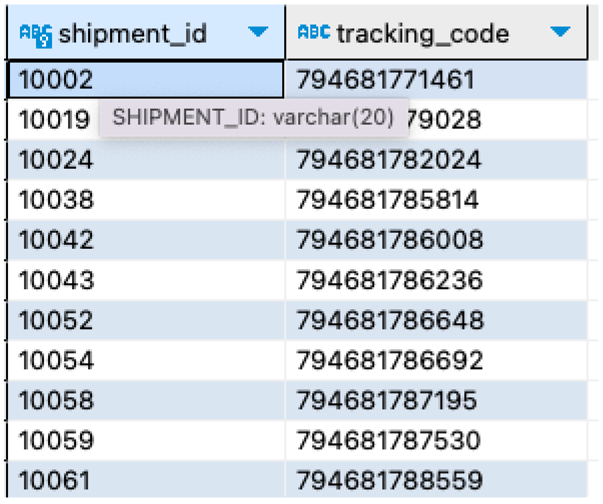
    - View or analyze shipments based on their unique tracking numbers. Each shipment is identified and tracked using a specific tracking number.

Solution :



```
select s.shipment_id, sprs.tracking_code
from shipment s
join shipment_package_route_seg sprs
using (shipment_id)
where sprs.TRACKING_CODE is not null
```

Output :

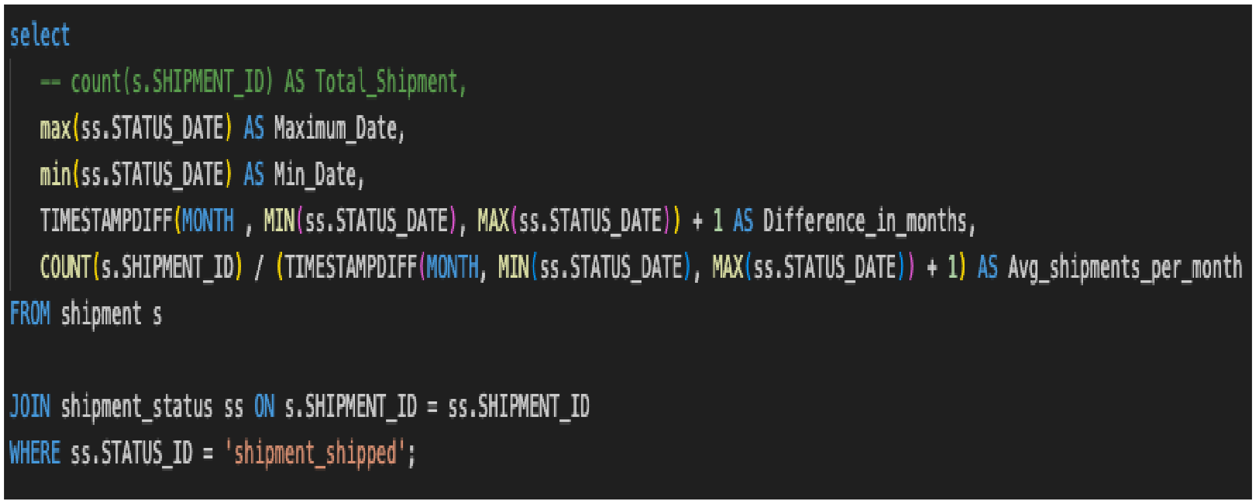


shipment_id	tracking_code
10002	794681771461
10019	79028
10024	794681782024
10038	794681785814
10042	794681786008
10043	794681786236
10052	794681786648
10054	794681786692
10058	794681787195
10059	794681787530
10061	794681788559

3. **Average number of shipments per month:**

   ○ Calculate the average number of shipments made per month by dividing the total number of shipments by the number of months.

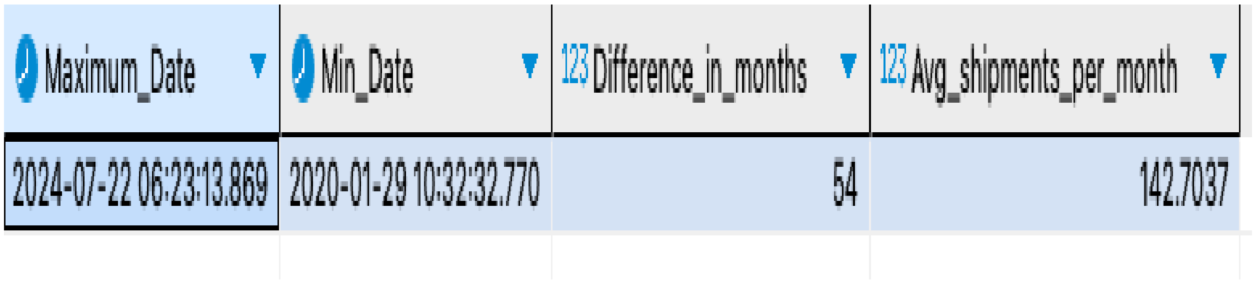
Solution :



```
select
    -- count(s.SHIPMENT_ID) AS Total_Shipment,
    max(ss.STATUS_DATE) AS Maximum_Date,
    min(ss.STATUS_DATE) AS Min_Date,
    TIMESTAMPDIFF(MONTH , MIN(ss.STATUS_DATE), MAX(ss.STATUS_DATE)) + 1 AS Difference_in_months,
    COUNT(s.SHIPMENT_ID) / (TIMESTAMPDIFF(MONTH, MIN(ss.STATUS_DATE), MAX(ss.STATUS_DATE)) + 1) AS Avg_shipments_per_month
FROM shipment s

JOIN shipment_status ss ON s.SHIPMENT_ID = ss.SHIPMENT_ID
WHERE ss.STATUS_ID = 'shipment_shipped';
```

Output :

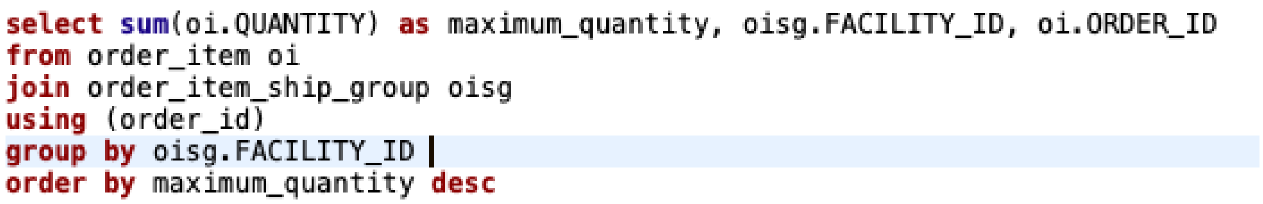


Maximum_Date	Min_Date	Difference_in_months	Avg_shipments_per_month
2024-07-22 06:23:13.869	2020-01-29 10:32:32.770	54	142.7037

4. Shipped units By Location:

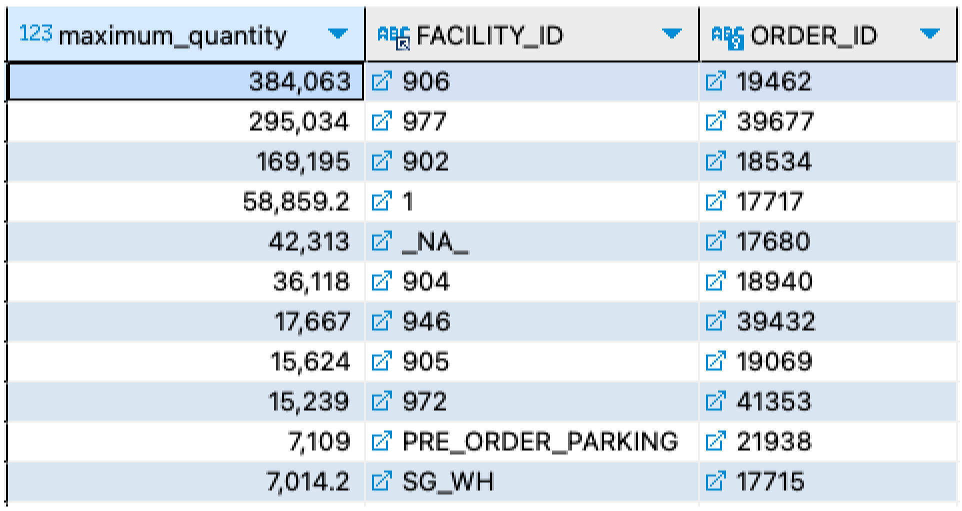
- Identify the number of units that have been shipped, categorized by different locations ;. Gain insights into the distribution of shipped units across various locations.

Solution:



```
select sum(oi.QUANTITY) as maximum_quantity, oisg.FACILITY_ID, oi.ORDER_ID
from order_item oi
join order_item_ship_group oisg
using (order_id)
group by oisg.FACILITY_ID
order by maximum_quantity desc
```

Output:

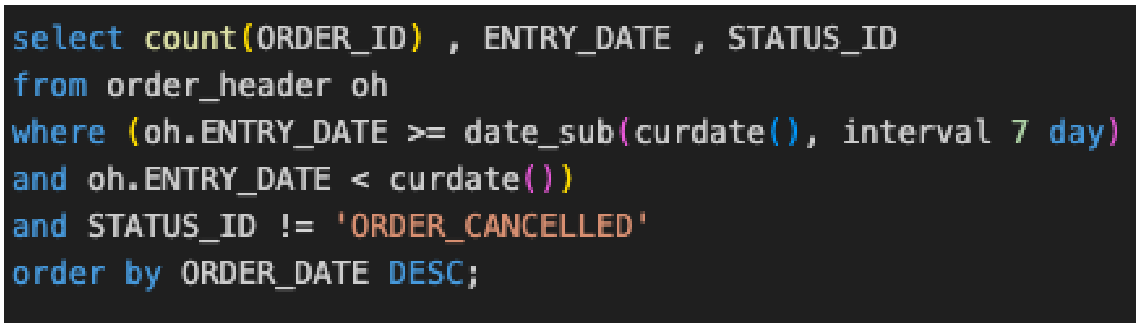


123 maximum_quantity	FACILITY_ID	ORDER_ID
384,063	906	19462
295,034	977	39677
169,195	902	18534
58,859.2	1	17717
42,313	_NA_	17680
36,118	904	18940
17,667	946	39432
15,624	905	19069
15,239	972	41353
7,109	PRE_ORDER_PARKING	21938
7,014.2	SG_WH	17715

5. Last week imported orders & items count:

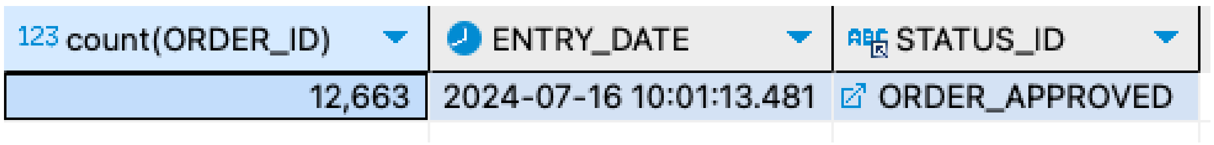
- Identify and count the orders and items that were imported in the system during the last week.

Solution :



```
select count(ORDER_ID) , ENTRY_DATE , STATUS_ID
from order_header oh
where (oh.ENTRY_DATE >= date_sub(curdate(), interval 7 day)
and oh.ENTRY_DATE < curdate())
and STATUS_ID != 'ORDER_CANCELLED'
order by ORDER_DATE DESC;
```

Output:

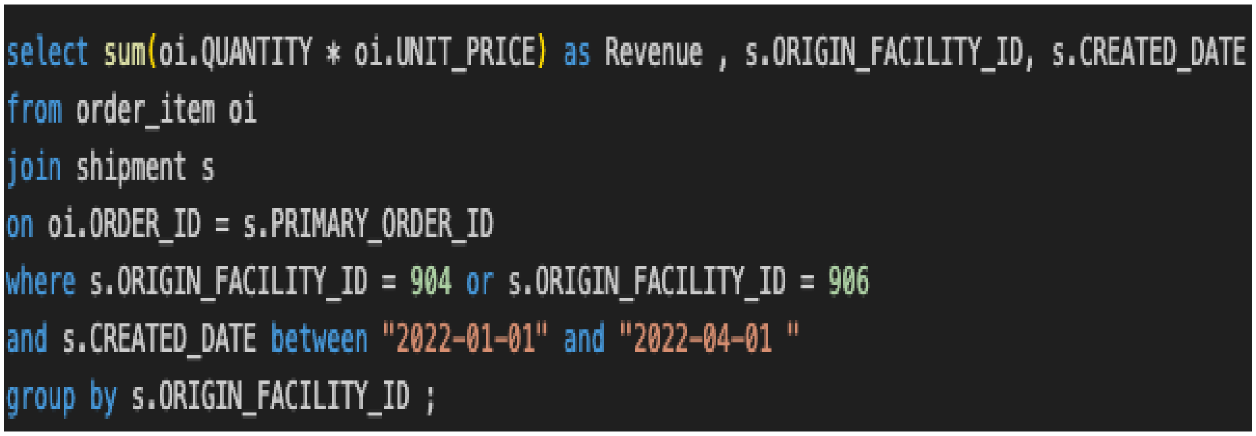


123 count(ORDER_ID)	ENTRY_DATE	STATUS_ID
12,663	2024-07-16 10:01:13.481	ORDER_APPROVED

6. Total $ value of shipments shipped from facility 904/906 to first quarter:

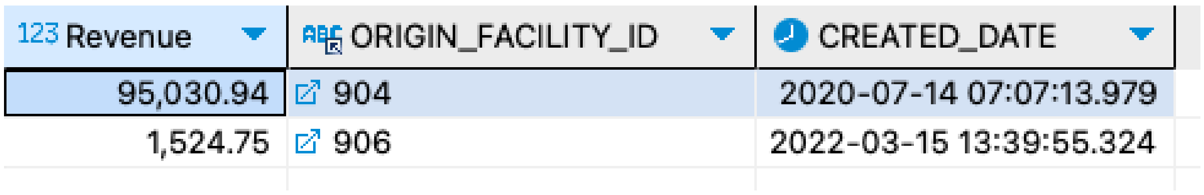
Calculate the total monetary value of shipments that originated from facilities 904 and 906 during the first quarter.

- Solution:



```
select sum(oi.QUANTITY * oi.UNIT_PRICE) as Revenue , s.ORIGIN_FACILITY_ID, s.CREATED_DATE
from order_item oi
join shipment s
on oi.ORDER_ID = s.PRIMARY_ORDER_ID
where s.ORIGIN_FACILITY_ID = 904 or s.ORIGIN_FACILITY_ID = 906
and s.CREATED_DATE between "2022-01-01" and "2022-04-01 "
group by s.ORIGIN_FACILITY_ID ;
```

Output:

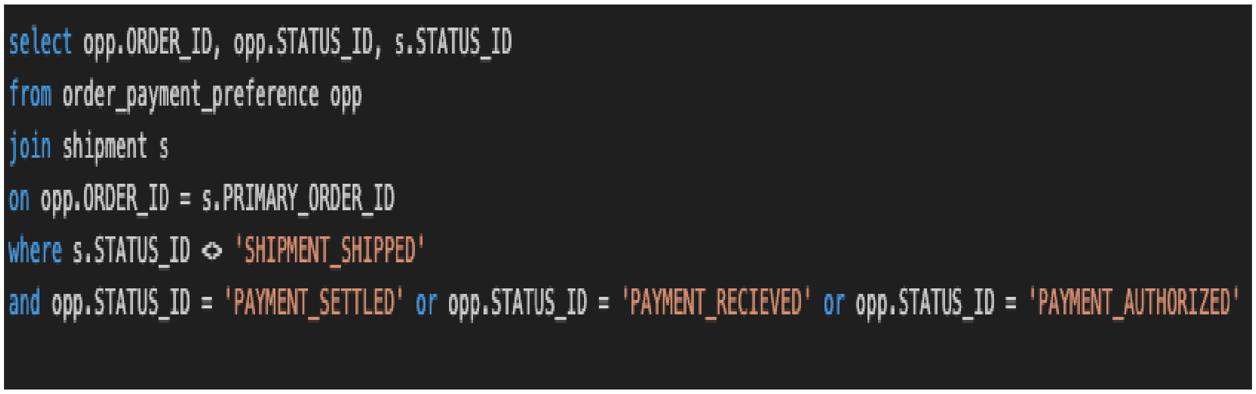


Revenue	ORIGIN_FACILITY_ID	CREATED_DATE
95,030.94	904	2020-07-14 07:07:13.979
1,524.75	906	2022-03-15 13:39:55.324

7. **Payment captured but not shipped order items:**

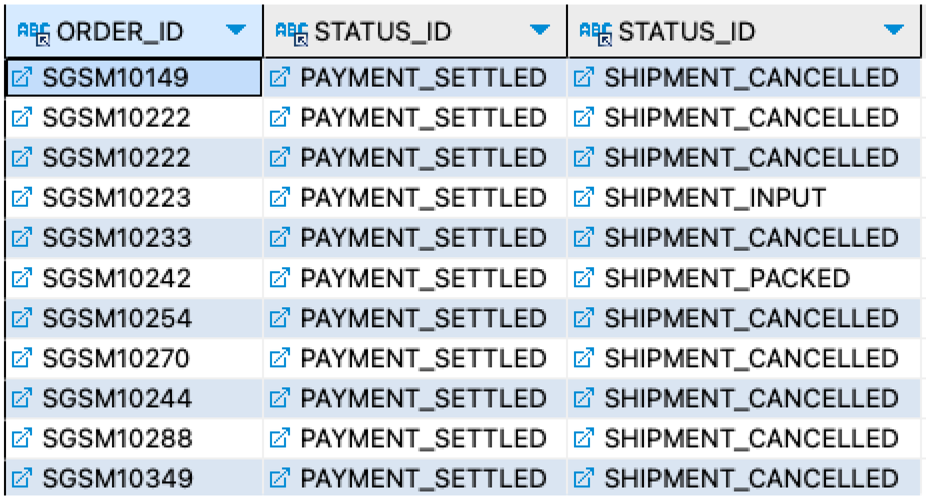
   - Identify orders where payment has been captured, but the items have not been shipped yet or shipment has not yet been created/initiated.

Solution :



```sql
select opp.ORDER_ID, opp.STATUS_ID, s.STATUS_ID
from order_payment_preference opp
join shipment s
on opp.ORDER_ID = s.PRIMARY_ORDER_ID
where s.STATUS_ID <> 'SHIPMENT_SHIPPED'
and opp.STATUS_ID = 'PAYMENT_SETTLED' or opp.STATUS_ID = 'PAYMENT_RECIEVED' or opp.STATUS_ID = 'PAYMENT_AUTHORIZED'
```

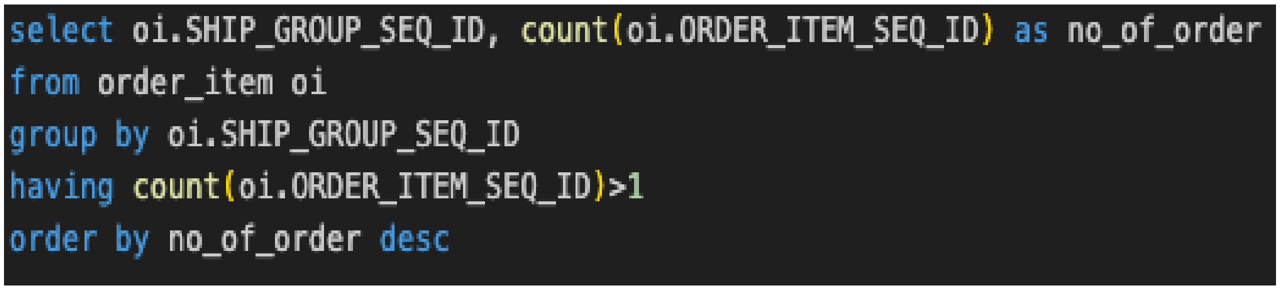
Output:



ORDER_ID	STATUS_ID	STATUS_ID
SGSM10149	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10222	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10222	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10223	PAYMENT_SETTLED	SHIPMENT_INPUT
SGSM10233	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10242	PAYMENT_SETTLED	SHIPMENT_PACKED
SGSM10254	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10270	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10244	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10288	PAYMENT_SETTLED	SHIPMENT_CANCELLED
SGSM10349	PAYMENT_SETTLED	SHIPMENT_CANCELLED

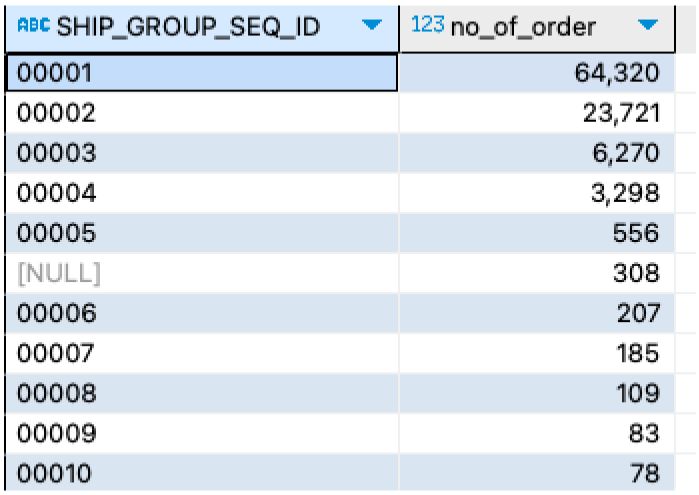
8. **Orders that have more than one item in a single ship group:**

Solution :



```sql
select oi.SHIP_GROUP_SEQ_ID, count(oi.ORDER_ITEM_SEQ_ID) as no_of_order
from order_item oi
group by oi.SHIP_GROUP_SEQ_ID
having count(oi.ORDER_ITEM_SEQ_ID)>1
order by no_of_order desc
```

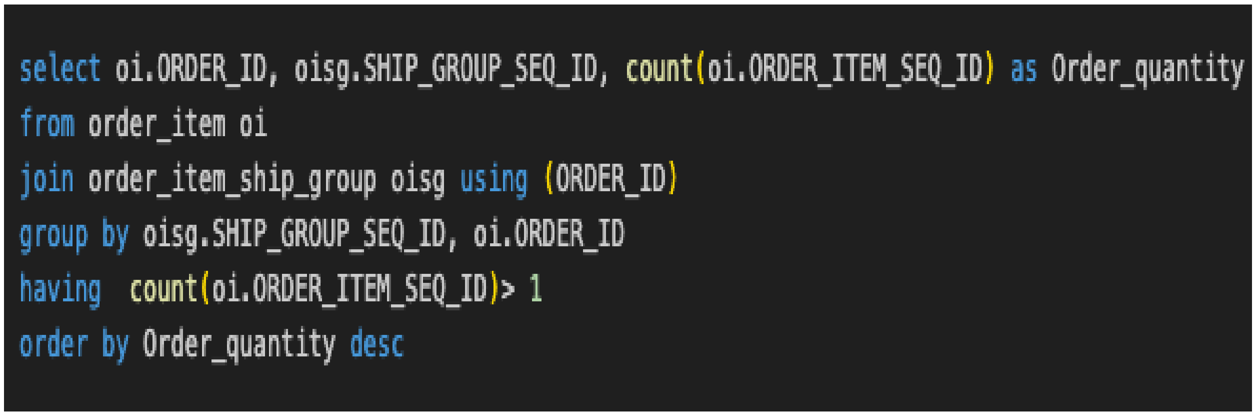
Output:



SHIP_GROUP_SEQ_ID	no_of_order
00001	64,320
00002	23,721
00003	6,270
00004	3,298
00005	556
[NULL]	308
00006	207
00007	185
00008	109
00009	83
00010	78

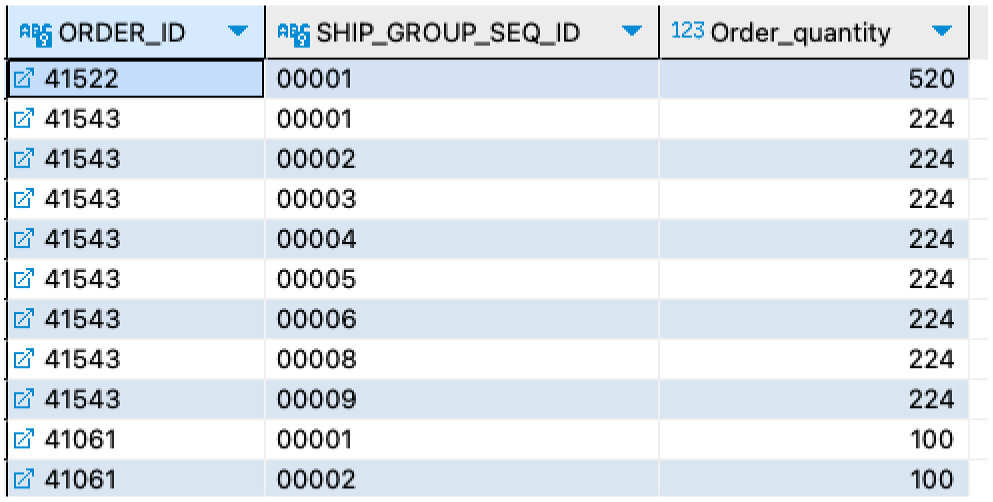
9. **Find orders where multiple items are grouped and shipped together in a single shipment:**

Solution:



```
select oi.ORDER_ID, oisg.SHIP_GROUP_SEQ_ID, count(oi.ORDER_ITEM_SEQ_ID) as Order_quantity
from order_item oi
join order_item_ship_group oisg using (ORDER_ID)
group by oisg.SHIP_GROUP_SEQ_ID, oi.ORDER_ID
having  count(oi.ORDER_ITEM_SEQ_ID)> 1
order by Order_quantity desc
```

Output:

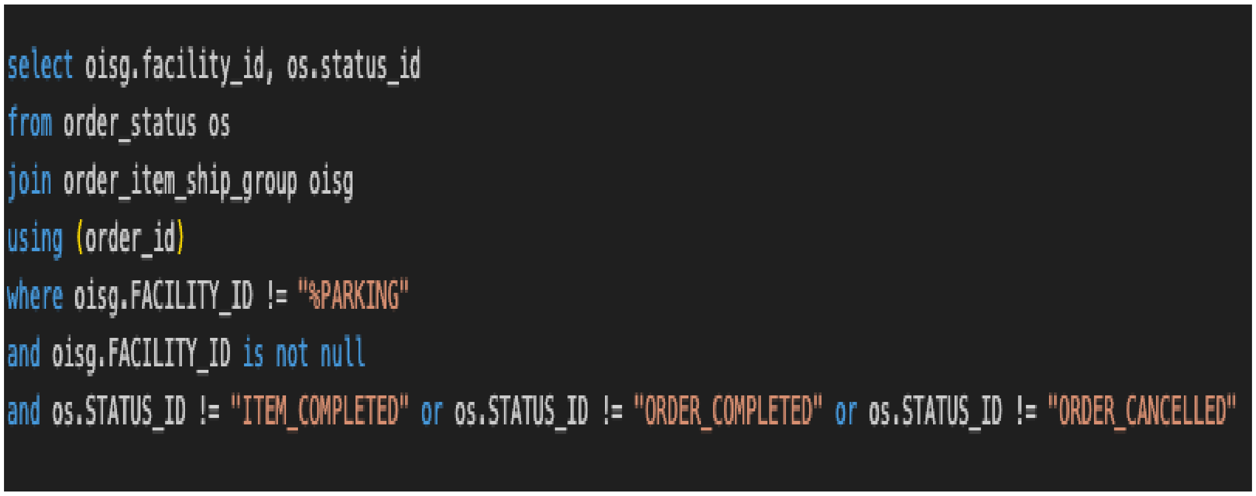


ORDER_ID	SHIP_GROUP_SEQ_ID	Order_quantity
41522	00001	520
41543	00001	224
41543	00002	224
41543	00003	224
41543	00004	224
41543	00005	224
41543	00006	224
41543	00008	224
41543	00009	224
41061	00001	100
41061	00002	100

10. **Orders brokered but not shipped:**

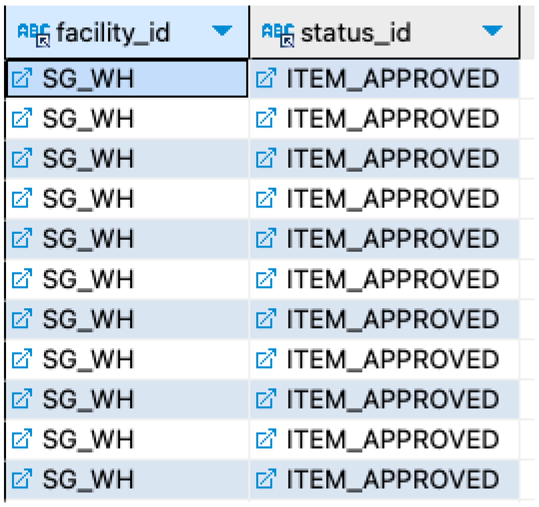
- Identify orders that have been brokered (arranged or negotiated) but have not been shipped yet or shipment has not yet been created/initiated.

Solution:



```
select oisg.facility_id, os.status_id
from order_status os
join order_item_ship_group oisg
using (order_id)
where oisg.FACILITY_ID != "%PARKING"
and oisg.FACILITY_ID is not null
and os.STATUS_ID != "ITEM_COMPLETED" or os.STATUS_ID != "ORDER_COMPLETED" or os.STATUS_ID != "ORDER_CANCELLED"
```

Output :

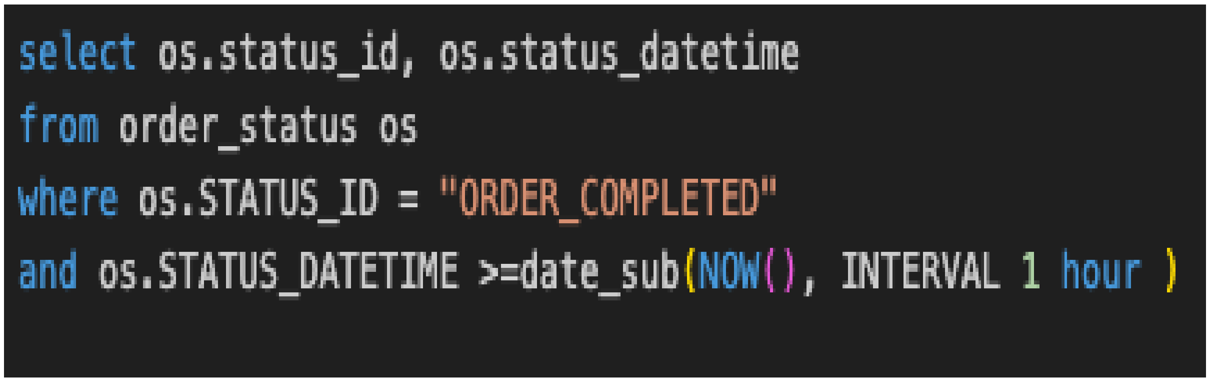


facility_id	status_id
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED
SG_WH	ITEM_APPROVED

11. **Orders completed hourly:**

- Analyze and present the distribution of completed orders on an hourly basis.

Solution:



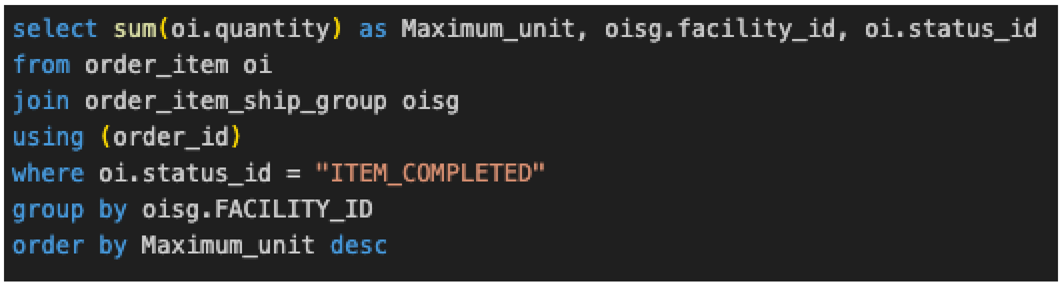
```
select os.status_id, os.status_datetime
from order_status os
where os.STATUS_ID = "ORDER_COMPLETED"
and os.STATUS_DATETIME >=date_sub(NOW(), INTERVAL 1 hour )
```

No Output

12. **Maximum units fulfilled by location:**

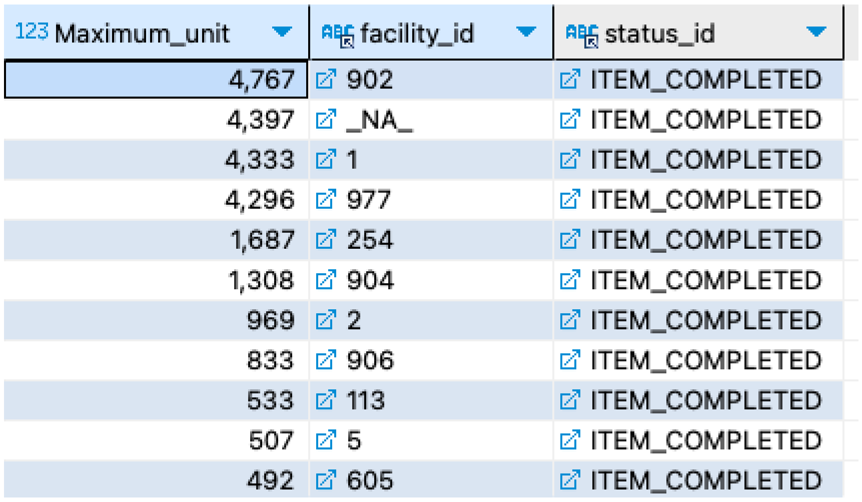
- Identify the location that has fulfilled the maximum number of units. This provides insights into the efficiency of different fulfillment centers.

Solution:



```sql
select sum(oi.quantity) as Maximum_unit, oisg.facility_id, oi.status_id
from order_item oi
join order_item_ship_group oisg
using (order_id)
where oi.status_id = "ITEM_COMPLETED"
group by oisg.FACILITY_ID
order by Maximum_unit desc
```

Output :

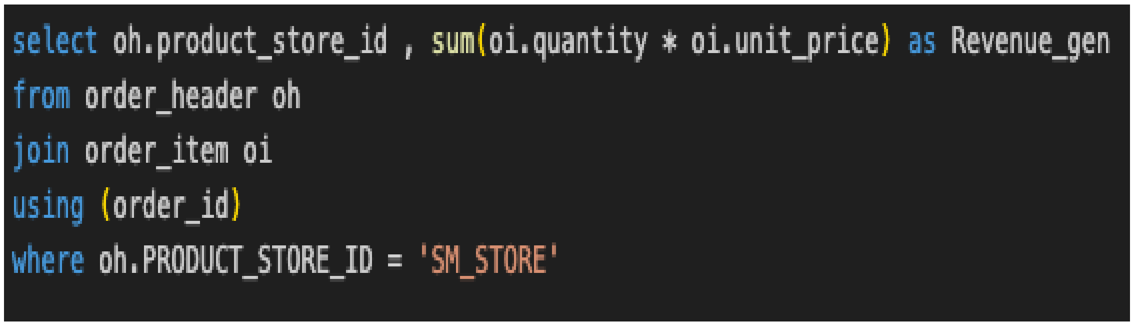


123 Maximum_unit	facility_id	status_id
4,767	902	ITEM_COMPLETED
4,397	_NA_	ITEM_COMPLETED
4,333	1	ITEM_COMPLETED
4,296	977	ITEM_COMPLETED
1,687	254	ITEM_COMPLETED
1,308	904	ITEM_COMPLETED
969	2	ITEM_COMPLETED
833	906	ITEM_COMPLETED
533	113	ITEM_COMPLETED
507	5	ITEM_COMPLETED
492	605	ITEM_COMPLETED

13. **facility wise Revenue for (SM Store):**

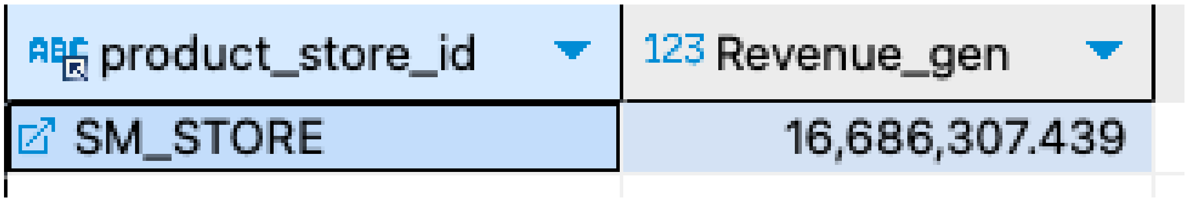
- Break down the revenue generated by each store. This helps in understanding the contribution of individual stores to the overall revenue.

Solution:



```
select oh.product_store_id , sum(oi.quantity * oi.unit_price) as Revenue_gen
from order_header oh
join order_item oi
using (order_id)
where oh.PRODUCT_STORE_ID = 'SM_STORE'
```

Output :

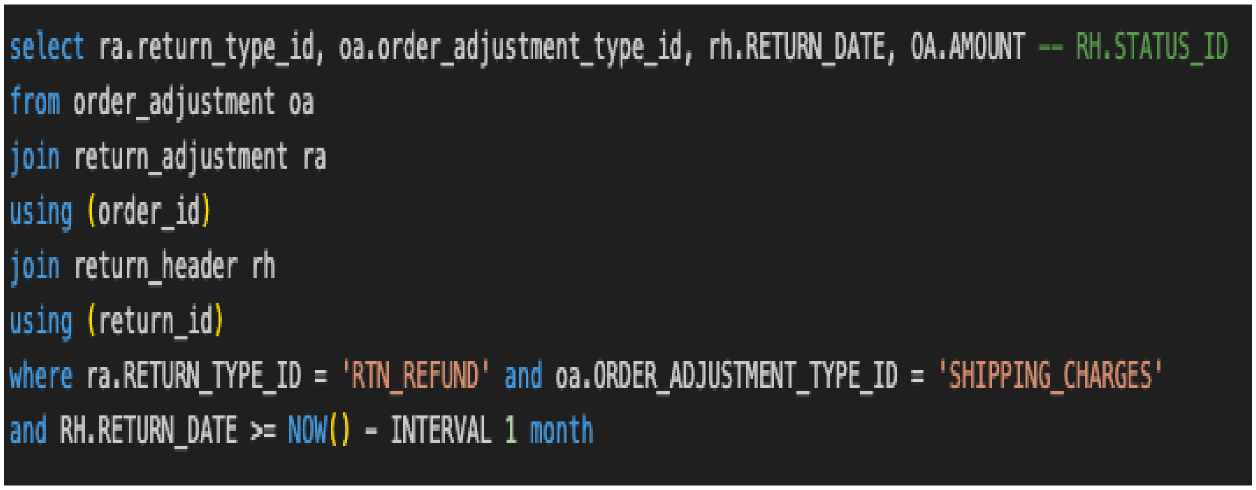


product_store_id	Revenue_gen
SM_STORE	16,686,307.439

14. **Shipping Refund in the last month:**

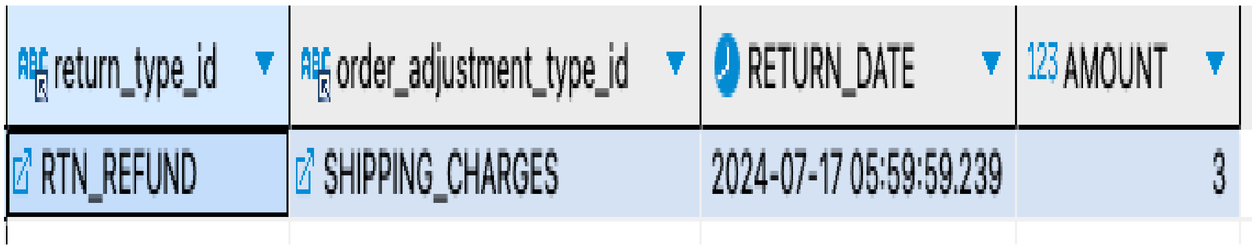
- Calculate the refunds issued specifically for shipping charges in the last month.

Solution:



```
select ra.return_type_id, oa.order_adjustment_type_id, rh.RETURN_DATE, OA.AMOUNT -- RH.STATUS_ID
from order_adjustment oa
join return_adjustment ra
using (order_id)
join return_header rh
using (return_id)
where ra.RETURN_TYPE_ID = 'RTN_REFUND' and oa.ORDER_ADJUSTMENT_TYPE_ID = 'SHIPPING_CHARGES'
and RH.RETURN_DATE >= NOW() - INTERVAL 1 month
```

Output :

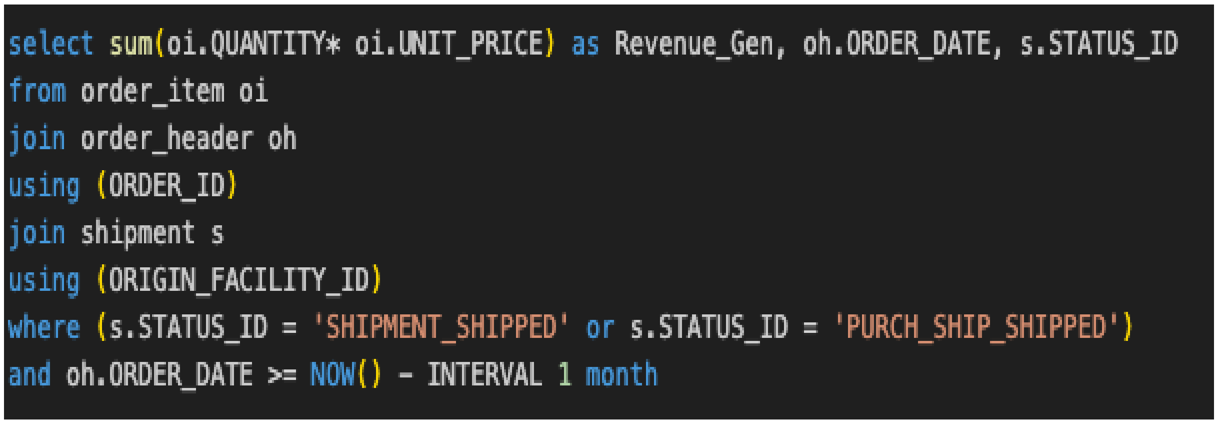


return_type_id	order_adjustment_type_id	RETURN_DATE	AMOUNT
RTN_REFUND	SHIPPING_CHARGES	2024-07-17 05:59:59.239	3

15. **Shipping Revenue last month:**

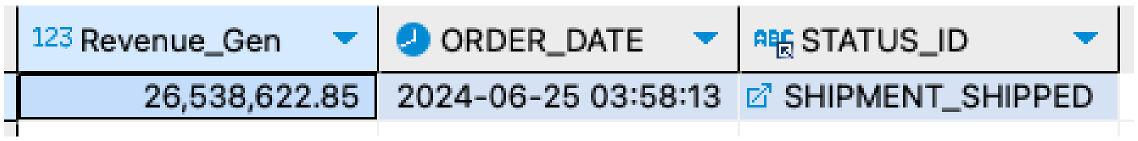
  - Determine the total revenue generated from shipping in the last month.

Solution:



```
select sum(oi.QUANTITY* oi.UNIT_PRICE) as Revenue_Gen, oh.ORDER_DATE, s.STATUS_ID
from order_item oi
join order_header oh
using (ORDER_ID)
join shipment s
using (ORIGIN_FACILITY_ID)
where (s.STATUS_ID = 'SHIPMENT_SHIPPED' or s.STATUS_ID = 'PURCH_SHIP_SHIPPED')
and oh.ORDER_DATE >= NOW() - INTERVAL 1 month
```

Output :

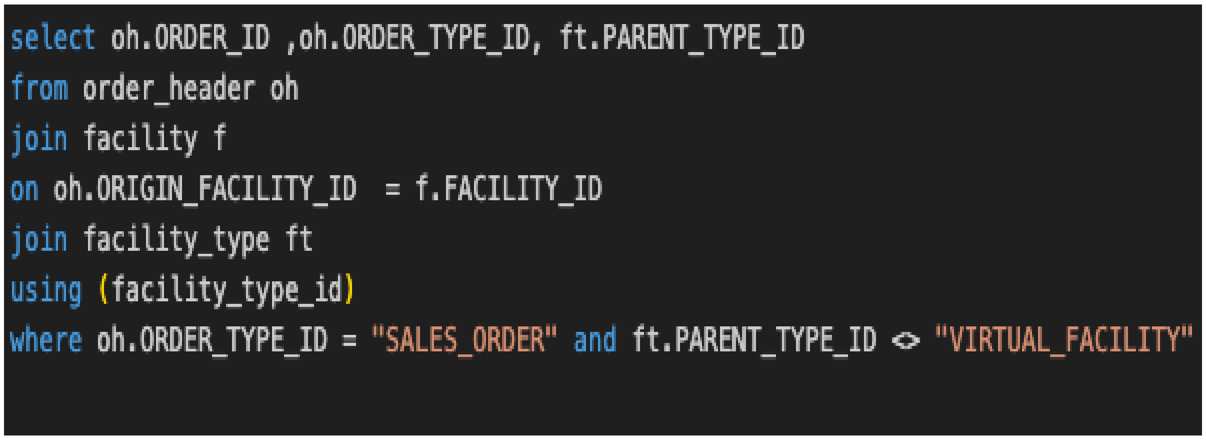


Revenue_Gen	ORDER_DATE	STATUS_ID
26,538,622.85	2024-06-25 03:58:13	SHIPMENT_SHIPPED

16. **Send sale orders shipped from the warehouse:**

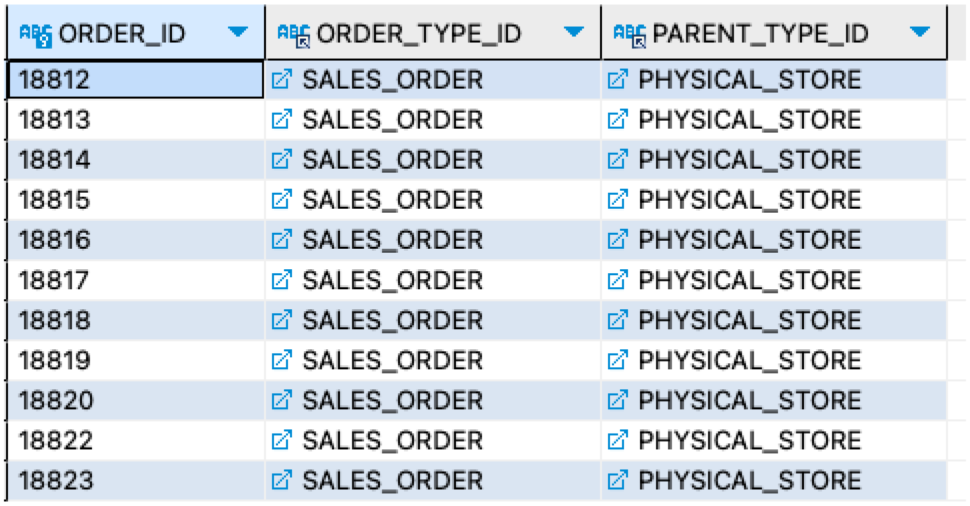
  - Identify send sale orders that have been shipped from the warehouse.

Solution:



```
select oh.ORDER_ID ,oh.ORDER_TYPE_ID, ft.PARENT_TYPE_ID
from order_header oh
join facility f
on oh.ORIGIN_FACILITY_ID  = f.FACILITY_ID
join facility_type ft
using (facility_type_id)
where oh.ORDER_TYPE_ID = "SALES_ORDER" and ft.PARENT_TYPE_ID <> "VIRTUAL_FACILITY"
```

Output :

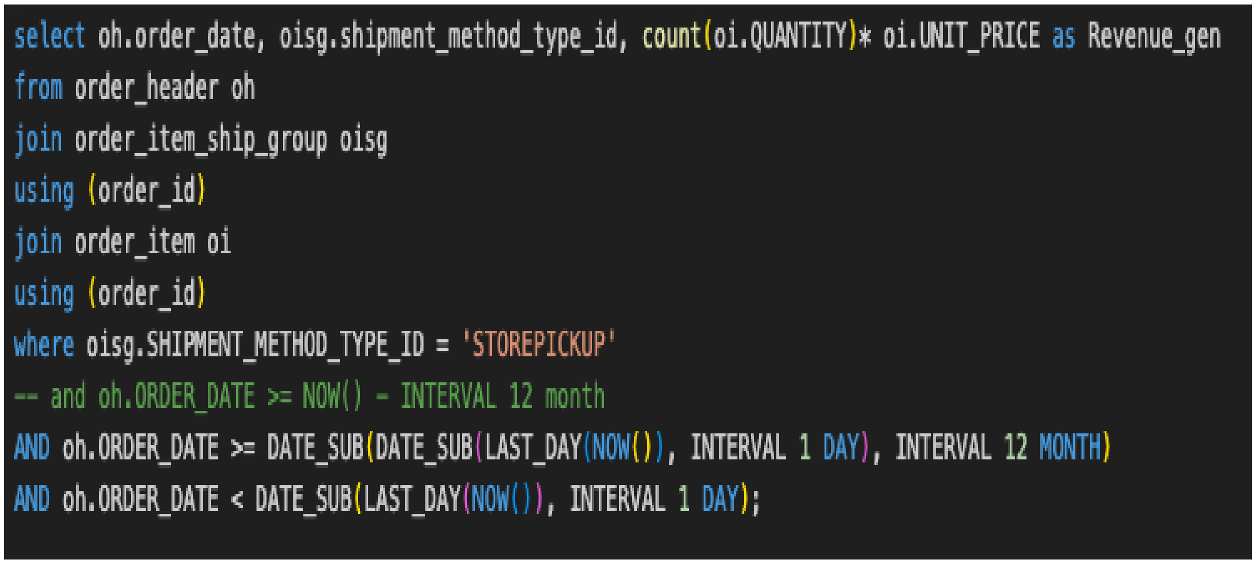


ORDER_ID	ORDER_TYPE_ID	PARENT_TYPE_ID
18812	SALES_ORDER	PHYSICAL_STORE
18813	SALES_ORDER	PHYSICAL_STORE
18814	SALES_ORDER	PHYSICAL_STORE
18815	SALES_ORDER	PHYSICAL_STORE
18816	SALES_ORDER	PHYSICAL_STORE
18817	SALES_ORDER	PHYSICAL_STORE
18818	SALES_ORDER	PHYSICAL_STORE
18819	SALES_ORDER	PHYSICAL_STORE
18820	SALES_ORDER	PHYSICAL_STORE
18822	SALES_ORDER	PHYSICAL_STORE
18823	SALES_ORDER	PHYSICAL_STORE

18. **BOPIS orders Revenue in the last year:**

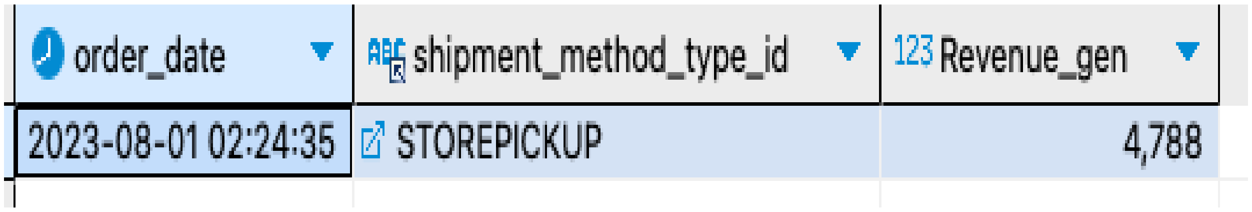
- Calculate the revenue generated from BOPIS orders over the past year.

Solution:



```
select oh.order_date, oisg.shipment_method_type_id, count(oi.QUANTITY)* oi.UNIT_PRICE as Revenue_gen
from order_header oh
join order_item_ship_group oisg
using (order_id)
join order_item oi
using (order_id)
where oisg.SHIPMENT_METHOD_TYPE_ID = 'STOREPICKUP'
-- and oh.ORDER_DATE >= NOW() - INTERVAL 12 month
AND oh.ORDER_DATE >= DATE_SUB(DATE_SUB(LAST_DAY(NOW()), INTERVAL 1 DAY), INTERVAL 12 MONTH)
AND oh.ORDER_DATE < DATE_SUB(LAST_DAY(NOW()), INTERVAL 1 DAY);
```

Output :



order_date	shipment_method_type_id	Revenue_gen
2023-08-01 02:24:35	STOREPICKUP	4,788