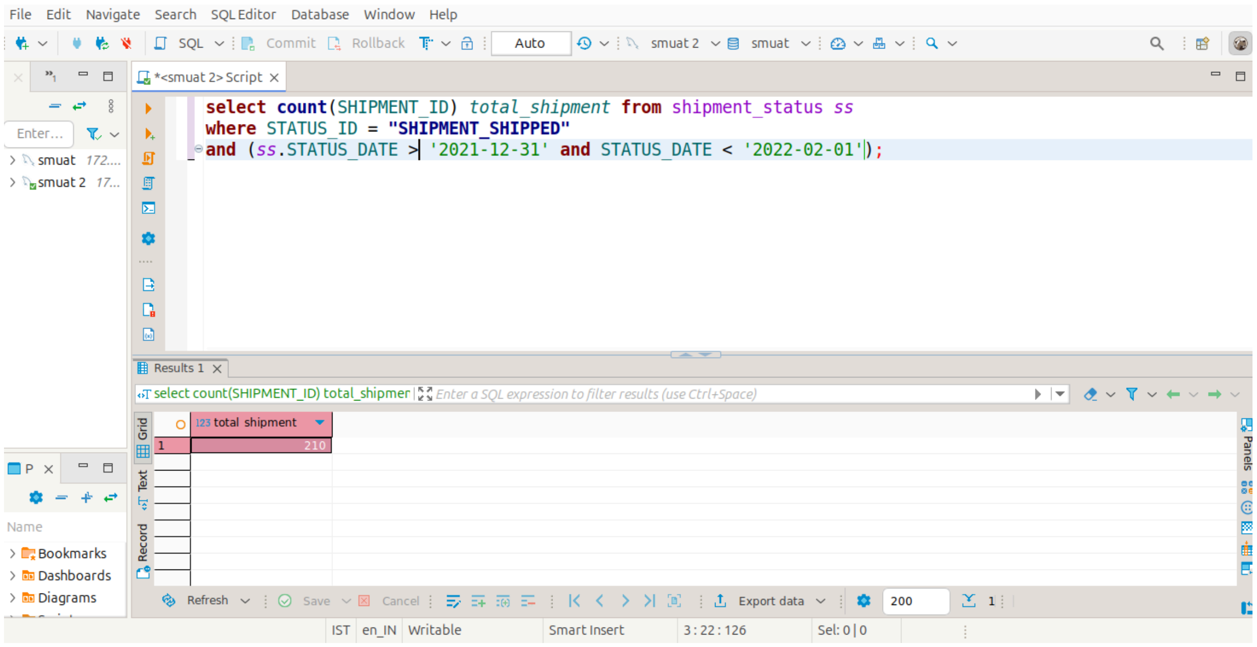
SQL ASSIGNMENT 1

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

Query:-

```
Select count(SHIPMENT_ID) Jan_shipment from      shipment_status ss
where STATUS_ID = "SHIPMENT_SHIPPED"
and (ss.STATUS_DATE >= '2021-12-31' and ss.STATUS_DATE <=
'2022-02-01');
```



```
select count(SHIPMENT_ID) total_shipment from shipment_status ss
where STATUS_ID = "SHIPMENT_SHIPPED"
and (ss.STATUS_DATE > '2021-12-31' and STATUS_DATE < '2022-02-01');
```

	total shipment
1	210

Explanation :-

The SQL query counts how many shipments were shipped in January 2022. It looks at the `shipment_status` table and checks for entries where the `STATUS_ID` is "SHIPMENT_SHIPPED." It only counts those shipments whose `STATUS_DATE` is between December 31, 2021, and February 1, 2022. This helps to find out the total number of shipments made during January 2022. You can change the dates or add more filters if you want to get different information.

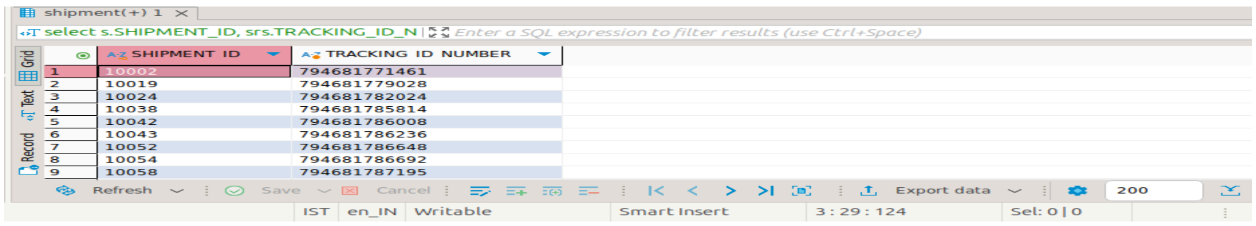
2. Shipment by Tracking number:

Query:-

```
select s.SHIPMENT_ID, srs.TRACKING_ID_NUMBER from
shipment s
join shipment_route_segment srs
on s.SHIPMENT_ID =srs.SHIPMENT_ID where
srs.TRACKING_ID_NUMBER  is not null
```

Explanation:-

This SQL query retrieves the `SHIPMENT_ID` and `TRACKING_ID_NUMBER` for shipments that have a tracking number associated with them. It does this by selecting data from two tables: `shipment` (aliased as `s`) and `shipment_route_segment` (aliased as `srs`). The query uses a JOIN to combine the two tables based on matching `SHIPMENT_ID` values. The condition `srs.TRACKING_ID_NUMBER is not null` ensures that only the shipments that have a valid tracking number are included in the results. This helps identify which shipments can be tracked.



shipment(+) 1

select s.SHIPMENT_ID, srs.TRACKING_ID_N

	SHIPMENT ID	TRACKING ID NUMBER
1	10002	794681771461
2	10019	794681779028
3	10024	794681782024
4	10038	794681785814
5	10042	794681786008
6	10043	794681786236
7	10052	794681786648
8	10054	794681786692
9	10058	794681787195

3. Average number of shipments per month:

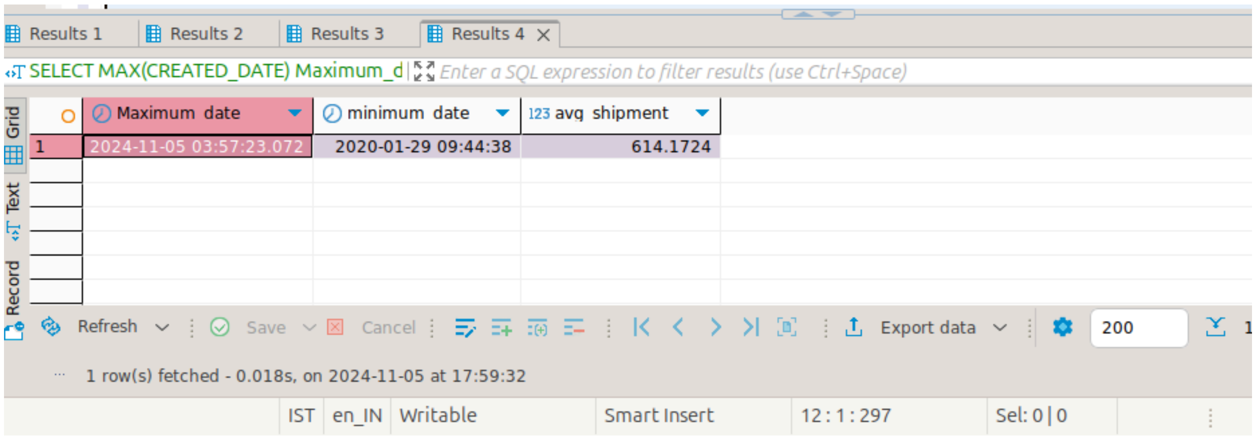
Query:-

```
SELECT
   -- max return latest shipment date
   MAX(CREATED_DATE) Maximum_date,
   -- min return oldest shipment date
   MIN(CREATED_DATE) minimum_date,
   COUNT(SHIPMENT_ID) / (TIMESTAMPDIFF(MONTH,
MIN(CREATED_DATE), MAX(CREATED_DATE)) + 1) AS
"avg_shipment"
```

```
FROM
    shipment s;
```

Explanation:-

This SQL query calculates important details about shipments from a table called `shipment`. It finds the most recent shipment date and the oldest shipment date. It also counts how many shipments there are in total. Then, it calculates the number of months between the earliest and latest shipment dates and adds one to include both months. Finally, it divides the total number of shipments by the number of months to determine the average number of shipments per month. The results include the maximum date, minimum date, and the average shipments per month.

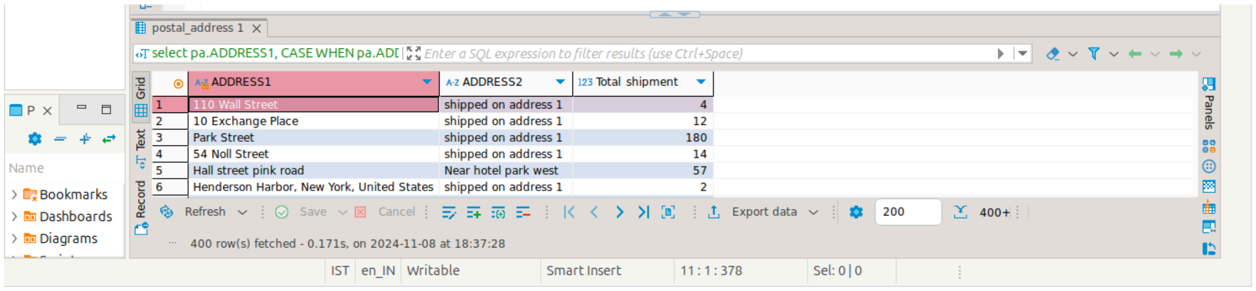


```
SELECT MAX(CREATED_DATE) Maximum_d
```

	Maximum date	minimum date	avg shipment
1	2024-11-05 03:57:23.072	2020-01-29 09:44:38	614.1724

```
1 row(s) fetched - 0.018s, on 2024-11-05 at 17:59:32
IST  en_IN  Writable  Smart Insert  12 : 1 : 297  Sel: 0 | 0
```

4. Shipped units By Location:



```
select pa.ADDRESS1, CASE WHEN pa.ADI
```

	ADDRESS1	ADDRESS2	Total shipment
1	110 Wall Street	shipped on address 1	4
2	10 Exchange Place	shipped on address 1	12
3	Park Street	shipped on address 1	180
4	54 Noll Street	shipped on address 1	14
5	Hall street pink road	Near hotel park west	57
6	Henderson Harbor, New York, United States	shipped on address 1	2

```
400 row(s) fetched - 0.171s, on 2024-11-08 at 18:37:28
IST  en_IN  Writable  Smart Insert  11 : 1 : 378  Sel: 0 | 0
```

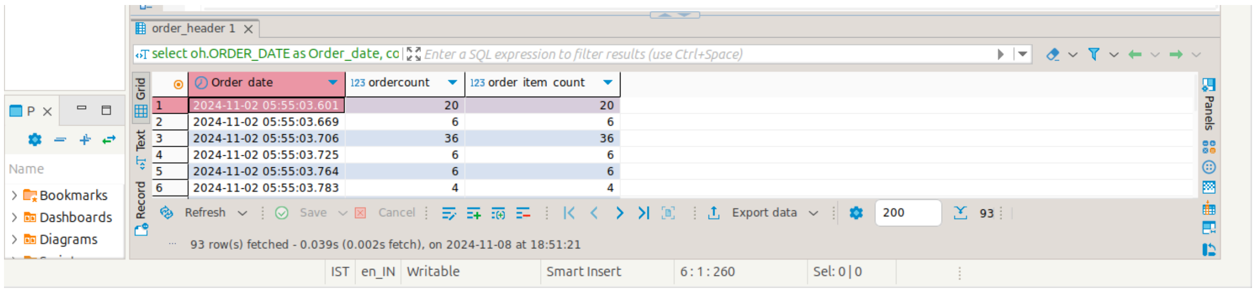
Query:-

```sql
select pa.POSTAL_CODE,
CASE WHEN pa.POSTAL_CODE is NULL
THEN 'check with'
ELSE pa.ADDRESS1
END AS 'ADDRESS1',
count(S.SHIPMENT_ID)
as Total_shipment
from shipment s join postal_address pa on
s.DESTINATION_CONTACT_MECH_ID = pa.CONTACT_MECH_ID
where s.STATUS_ID in (select STATUS_ID from
shipment_status ss where ss.STATUS_ID =
'shipment_shipped')
group by pa.POSTAL_CODE
```

Explanation:-

This SQL query retrieves shipment details based on specific addresses and statuses. It joins the shipment and postal_address tables, focusing on the primary address (ADDRESS1) and a conditional secondary address (ADDRESS2). If ADDRESS2 is null, it defaults to "shipped on address 1." The query counts the total shipments (SHIPMENT_ID) for each address combination, filtering by shipments with the status 'shipment_shipped.' The results are grouped by ADDRESS1, showing the total shipments for each primary address.

5. Last week's imported orders & items count



order_header 1

select oh.ORDER_DATE as Order_date, co

	Order date	ordercount	order item count
1	2024-11-02 05:55:03.601	20	20
2	2024-11-02 05:55:03.669	6	6
3	2024-11-02 05:55:03.706	36	36
4	2024-11-02 05:55:03.725	6	6
5	2024-11-02 05:55:03.764	6	6
6	2024-11-02 05:55:03.783	4	4

```
93 row(s) fetched - 0.039s (0.002s fetch), on 2024-11-08 at 18:51:21
```

Query:-

```
select oh.ORDER_DATE as Order_date, count(oh.ORDER_ID)
as ordercount,
count(oi.ORDER_ITEM_SEQ_ID) as order_item_count from
order_header oh
join order_item oi on oh.ORDER_ID = oi.ORDER_ID
where oh.ORDER_DATE >= Now() - INTERVAL 7 day
group by oh.ORDER_DATE
```

Explanation:-

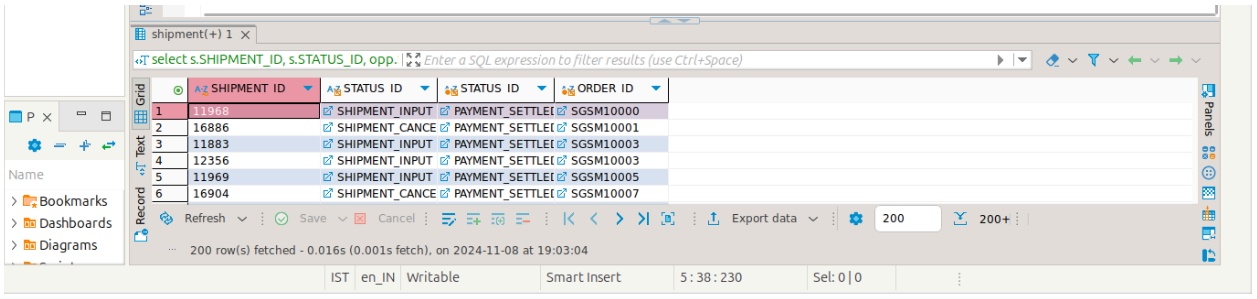
This SQL query counts the total number of orders and order items created in the last 7 days. It joins the order_header and order_item tables using ORDER_ID, filters the data to include orders from the past week, and counts the orders and items for each day. The results are grouped by ORDER_DATE, providing a daily summary of the number of orders and items created during the past week.

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

Query:-

Explanation:-

7. Payment captured but not shipped order items:-



shipment(+) 1

select s.SHIPMENT_ID, s.STATUS_ID, opp.

	SHIPMENT ID	STATUS ID	STATUS ID	ORDER ID
1	11968	SHIPMENT_INPUT	PAYMENT_SETTLEI	SGSM10000
2	16886	SHIPMENT_CANCE	PAYMENT_SETTLEI	SGSM10001
3	11883	SHIPMENT_INPUT	PAYMENT_SETTLEI	SGSM10003
4	12356	SHIPMENT_INPUT	PAYMENT_SETTLEI	SGSM10003
5	11969	SHIPMENT_INPUT	PAYMENT_SETTLEI	SGSM10005
6	16904	SHIPMENT_CANCE	PAYMENT_SETTLEI	SGSM10007

200 row(s) fetched - 0.016s (0.001s fetch), on 2024-11-08 at 19:03:04

Query:-

```
select s.SHIPMENT_ID, s.STATUS_ID, opp.STATUS_ID,
opp.ORDER_ID
from shipment s join order_payment_preference opp
on s.PRIMARY_ORDER_ID =opp.ORDER_ID
where opp.STATUS_ID = 'payment_settled'
and s.STATUS_ID != 'shipment_shipped'
```

Explanation:-

This SQL query retrieves shipment details by joining the shipment and order_payment_preference tables using the PRIMARY_ORDER_ID and ORDER_ID fields. It filters the results to show shipments where the associated order has a payment status of 'payment_settled' and the shipment's status is not 'shipment_shipped'. The query returns the SHIPMENT_ID, STATUS_ID from both the shipment and order_payment_preference tables, along with the ORDER_ID.

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

Query:-

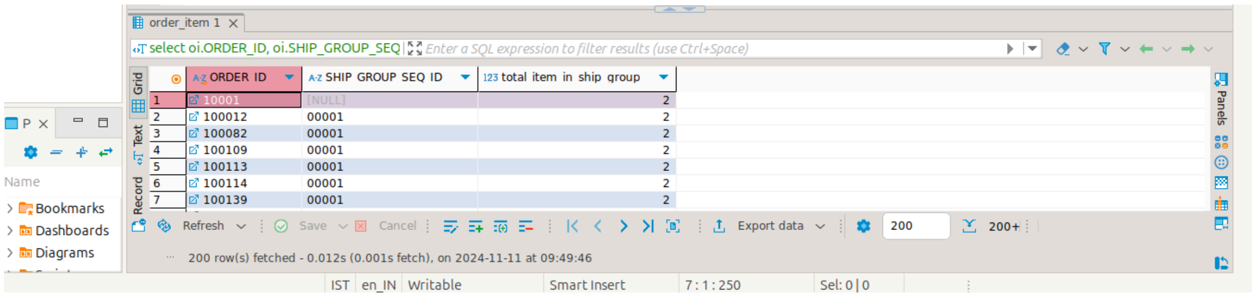
```
select oi.ORDER_ID, oi.SHIP_GROUP_SEQ_ID, count(oi.ORDER_ITEM_SEQ_ID) as
total_item_in_ship_group

from order_item oi where oi.ORDER_ITEM_SEQ_ID is not null

group by oi.ORDER_ID, oi.ORDER_ITEM_GROUP_SEQ_ID
having total_item_in_ship_group  > 1 ;
```

Explanation:-

This query finds orders with multiple items in each shipping group by counting items in each group and showing only those with more than one item.



```
order_item 1
select oi.ORDER_ID, oi.SHIP_GROUP_SEQ
```

	ORDER ID	SHIP GROUP SEQ ID	total item in ship group
1	10001	[NULL]	2
2	100012	00001	2
3	100082	00001	2
4	100109	00001	2
5	100113	00001	2
6	100114	00001	2
7	100139	00001	2

```
200 row(s) fetched - 0.012s (0.001s fetch), on 2024-11-11 at 09:49:46
IST  en_IN  Writable  Smart Insert  7 : 1 : 250  Sel: 0 | 0
```

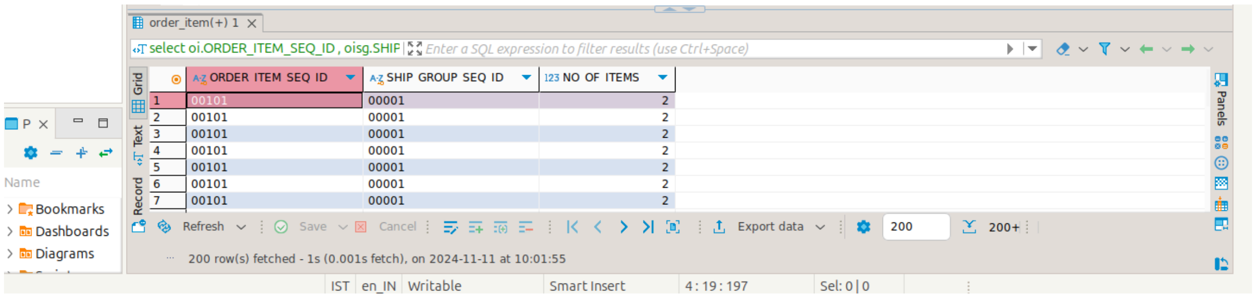
9. Find orders where multiple items are grouped and shipped together in a single shipment:

Query:-

```sql
select oi.ORDER_ITEM_SEQ_ID , oisg.SHIP_GROUP_SEQ_ID,
count(oi.ORDER_ITEM_SEQ_ID) as NO_OF_ITEMS
from order_item oi join order_item_ship_group oisg on oi.ORDER_ID =
oisg.ORDER_ID
group by oi.ORDER_ID , oisg.SHIP_GROUP_SEQ_ID
having NO_OF_ITEMS > 1 order by oi.ORDER_ID ;
```

Explanation:-

This query identifies orders with more than one item in each shipping group. It joins order_item (oi) with order_item_ship_group (oisg) based on ORDER_ID, groups the results by ORDER_ID and SHIP_GROUP_SEQ_ID, and counts the items per shipping group. Only groups with more than one item are shown, sorted by ORDER_ID.



```
order_item(+) 1
select oi.ORDER_ITEM_SEQ_ID , oisg.SHIF
```

	ORDER ITEM SEQ ID	SHIP GROUP SEQ ID	NO OF ITEMS
1	00101	00001	2
2	00101	00001	2
3	00101	00001	2
4	00101	00001	2
5	00101	00001	2
6	00101	00001	2
7	00101	00001	2

```
200 row(s) fetched - 1s (0.001s fetch), on 2024-11-11 at 10:01:55
IST  en_IN  Writable  Smart Insert  4 : 19 : 197  Sel: 0 | 0
```

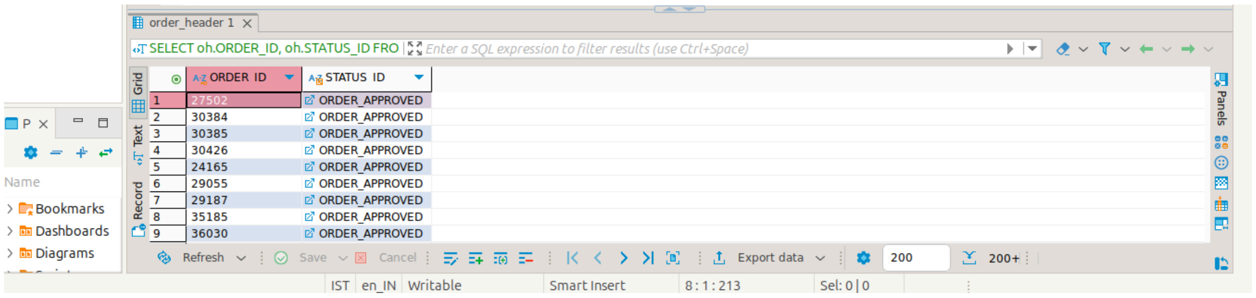
10. Orders brokered but not shipped:

Query:-

```
SELECT oh.ORDER_ID, oh.STATUS_ID
FROM order_header oh
JOIN shipment s ON oh.ORDER_ID = s.PRIMARY_ORDER_ID
WHERE oh.STATUS_ID = 'order_approved' AND s.STATUS_ID IS NOT null
group by oh.ORDER_ID , s.STATUS_ID ;
```

Explanation:-

This query retrieves approved orders and their statuses for orders with shipments that have a non-null status. It joins order_header with shipment, filters for approved orders, and groups by order and shipment status.



SELECT oh.ORDER_ID, oh.STATUS_ID FRO

	ORDER_ID	STATUS_ID
1	27502	ORDER_APPROVED
2	30384	ORDER_APPROVED
3	30385	ORDER_APPROVED
4	30426	ORDER_APPROVED
5	24165	ORDER_APPROVED
6	29055	ORDER_APPROVED
7	29187	ORDER_APPROVED
8	35185	ORDER_APPROVED
9	36030	ORDER_APPROVED

```
IST | en_IN | Writable | Smart Insert | 8 : 1 : 213 | Sel: 0 | 0
```

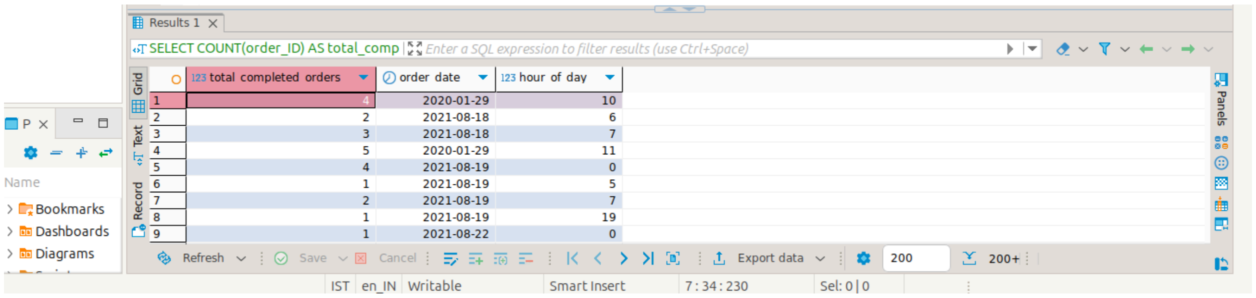
11. Orders completed hourly:

Query:-

```
SELECT
  COUNT(order_ID) AS total_completed_orders,
  DATE(STATUS_DATETIME) AS order_date,
  HOUR(STATUS_DATETIME) AS hour_of_day
FROM order_status os
WHERE os.STATUS_ID = 'order_completed'
GROUP BY order_date, hour_of_day;
```

Explanation:-

This query counts the number of completed orders (STATUS_ID = 'order_completed') for each hour on each specific date. It groups the results by both the date and the hour extracted from the STATUS_DATETIME field, providing the total number of completed orders for every hour of every day. This helps analyze the order completion trends by hour and day.



SELECT COUNT(order_ID) AS total_comp

	total completed orders	order date	hour of day
1	4	2020-01-29	10
2	2	2021-08-18	6
3	3	2021-08-18	7
4	5	2020-01-29	11
5	4	2021-08-19	0
6	1	2021-08-19	5
7	2	2021-08-19	7
8	1	2021-08-19	19
9	1	2021-08-22	0

IST en_IN Writable Smart Insert 7 : 34 : 230 Sel: 0 | 0

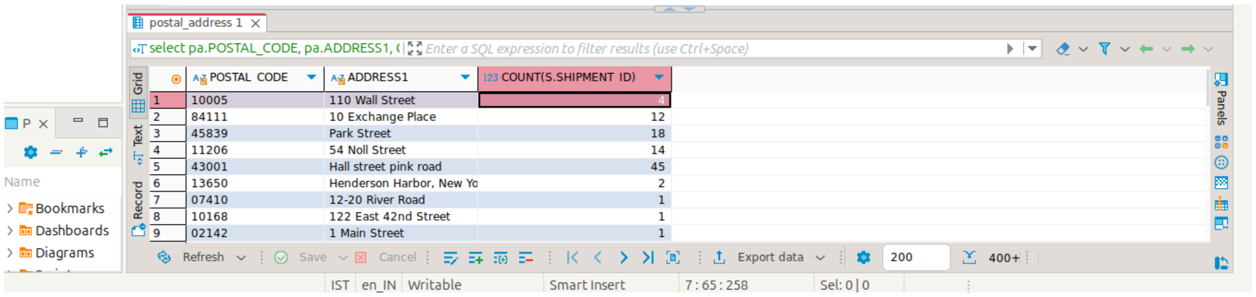
12. Maximum units fulfilled by location:

Query:-

```
select pa.POSTAL_CODE, pa.ADDRESS1,
COUNT(S.SHIPMENT_ID)
from shipment s
join postal_address pa
on s.DESTINATION_CONTACT_MECH_ID = pa.CONTACT_MECH_ID
where S.STATUS_ID = 'SHIPMENT_SHIPPED '
group by pa.POSTAL_CODE, pa.ADDRESS1 order by s.SHIPMENT_ID desc;
```

Explanation:-

This query retrieves the postal code and address of shipments that have been shipped (STATUS_ID = 'SHIPMENT_SHIPPED'). It joins the shipment table (s) with the postal_address table (pa) on the DESTINATION_CONTACT_MECH_ID and CONTACT_MECH_ID fields. The query counts the number of shipments (SHIPMENT_ID) for each postal code and address, groups the results by POSTAL_CODE and ADDRESS1, and orders the results by SHIPMENT_ID in descending order. This helps analyze shipment distribution by postal code and address, with a focus on shipped orders.



postal_address 1

select pa.POSTAL_CODE, pa.ADDRESS1, (

	POSTAL_CODE	ADDRESS1	COUNT(S.SHIPMENT_ID)
1	10005	110 Wall Street	4
2	84111	10 Exchange Place	12
3	45839	Park Street	18
4	11206	54 Noll Street	14
5	43001	Hall street pink road	45
6	13650	Henderson Harbor, New Yo	2
7	07410	12-20 River Road	1
8	10168	122 East 42nd Street	1
9	02142	1 Main Street	1

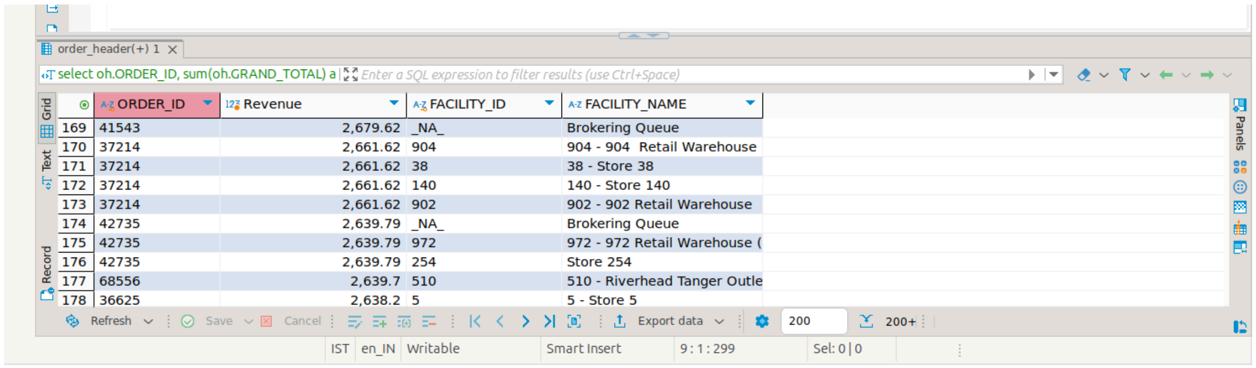
IST en_IN Writable Smart Insert 7 : 65 : 258 Sel: 0 | 0

13. facility wise Revenue for (SM Store):

Update Query:-

```
select oh.ORDER_ID, sum(oh.GRAND_TOTAL) as Revenue,
f.FACILITY_ID,
f.FACILITY_NAME
from order_header oh join
order_item_ship_group oisg on oh.ORDER_ID = oisg.ORDER_ID
join facility f on f.FACILITY_ID = oisg.FACILITY_ID
group by oh.ORDER_ID , f.FACILITY_ID , f.FACILITY_NAME
order by Revenue desc;
```

Result:-



order_header(+) 1

select oh.ORDER_ID, sum(oh.GRAND_TOTAL) a

	ORDER_ID	Revenue	FACILITY_ID	FACILITY_NAME
169	41543	2,679.62	_NA_	Brokering Queue
170	37214	2,661.62	904	904 - 904 Retail Warehouse
171	37214	2,661.62	38	38 - Store 38
172	37214	2,661.62	140	140 - Store 140
173	37214	2,661.62	902	902 - 902 Retail Warehouse
174	42735	2,639.79	_NA_	Brokering Queue
175	42735	2,639.79	972	972 - 972 Retail Warehouse (
176	42735	2,639.79	254	Store 254
177	68556	2,639.7	510	510 - Riverhead Tanger Outle
178	36625	2,638.2	5	5 - Store 5

IST en_IN Writable Smart Insert 9 : 1 : 299 Sel: 0 | 0

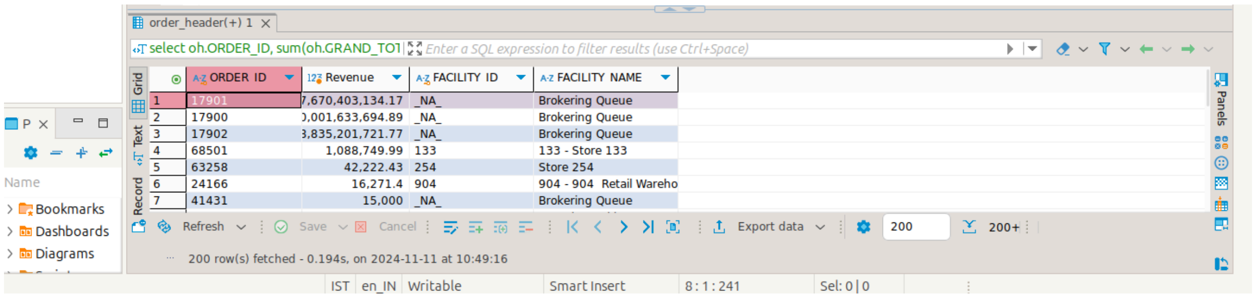
Query:-

```
select oh.ORDER_ID, sum(oh.GRAND_TOTAL) as Revenue,
f.FACILITY_ID,
f.FACILITY_NAME
from order_header oh join
facility f on oh.ORIGIN_FACILITY_ID = f.FACILITY_ID
```

```
group by oh.ORDER_ID , f.FACILITY_ID , f.FACILITY_NAME
order by Revenue desc;
```

Explanation:-

This query calculates the total revenue for each order by summing the GRAND_TOTAL from the order_header table (oh). It joins the order_header table with the facility table (f) based on the ORIGIN_FACILITY_ID and FACILITY_ID. The query groups the results by ORDER_ID, FACILITY_ID, and FACILITY_NAME, and orders the results by total revenue (Revenue) in descending order. This helps analyze the revenue generated from each order and the associated facility.



order_header(+) 1

select oh.ORDER_ID, sum(oh.GRAND_TOT

	ORDER ID	Revenue	FACILITY ID	FACILITY NAME
1	17901	7,670,403,134.17	_NA_	Brokering Queue
2	17900	),001,633,694.89	_NA_	Brokering Queue
3	17902	3,835,201,721.77	_NA_	Brokering Queue
4	68501	1,088,749.99	133	133 - Store 133
5	63258	42,222.43	254	Store 254
6	24166	16,271.4	904	904 - 904 Retail Wareho
7	41431	15,000	_NA_	Brokering Queue

200 row(s) fetched - 0.194s, on 2024-11-11 at 10:49:16

IST en_IN Writable Smart Insert 8 : 1 : 241 Sel: 0 | 0

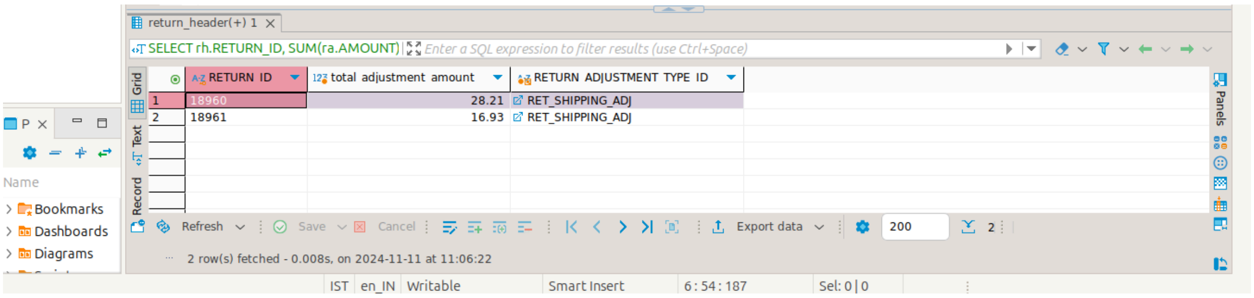
14. Shipping Refund in the last month:

Query:-

```
SELECT
  rh.RETURN_ID,
  SUM(ra.AMOUNT) AS total_adjustment_amount,
  ra.RETURN_ADJUSTMENT_TYPE_ID
FROM return_adjustment ra
JOIN return_header rh ON rh.RETURN_ID = ra.RETURN_ID
WHERE ra.RETURN_ADJUSTMENT_TYPE_ID = 'ret_shipping_adj'
AND rh.ENTRY_DATE >= NOW() - INTERVAL 1 MONTH
GROUP BY rh.RETURN_ID, ra.RETURN_ADJUSTMENT_TYPE_ID;
```

Explanation:-

This query calculates the total adjustment amount (`SUM(ra.AMOUNT)`) for returns with the adjustment type `'ret_shipping_adj'` from the `return_adjustment` table. It joins the `return_adjustment` table with the `return_header` table on `RETURN_ID`. The query filters for returns that were created in the last month (`rh.ENTRY_DATE >= NOW() - INTERVAL 1 MONTH`) and groups the results by `RETURN_ID` and `RETURN_ADJUSTMENT_TYPE_ID`. This helps analyze the total shipping adjustment amounts for each return within the last month.



return_header(+) 1

SELECT rh.RETURN_ID, SUM(ra.AMOUNT)

	RETURN ID	total adjustment amount	RETURN ADJUSTMENT TYPE ID
1	18960	28.21	RET_SHIPPING_ADJ
2	18961	16.93	RET_SHIPPING_ADJ

2 row(s) fetched - 0.008s, on 2024-11-11 at 11:06:22

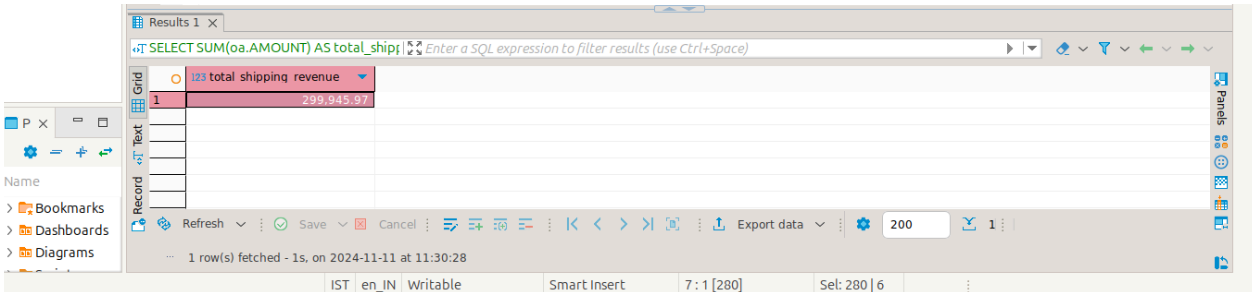
15. Shipping Revenue last month:

Query:-'

```
SELECT
  SUM(oa.AMOUNT) AS total_shipping_revenue
FROM order_adjustment oa
JOIN order_header oh ON oh.ORDER_ID = oa.ORDER_ID
WHERE oa.ORDER_ADJUSTMENT_TYPE_ID = 'shipping_charge' or
oa.ORDER_ADJUSTMENT_TYPE_ID = 'sales_tax'
AND oh.ENTRY_DATE >= CURDATE() - INTERVAL 1 MONTH;
```

Explanation:-

This query calculates the total revenue from shipping_charge and sales_tax adjustments in the last month. It sums the AMOUNT from the order_adjustment table, considering only the adjustments of type 'shipping_charge' or 'sales_tax'. The query joins the order_adjustment table with the order_header table on ORDER_ID and filters the results to include only those records where the ENTRY_DATE is within the last month. The result is a total of the shipping and sales tax adjustments made in the last month.



```
SELECT SUM(oa.AMOUNT) AS total_shipp

   total shipping revenue
1              299,945.97

1 row(s) fetched - 1s, on 2024-11-11 at 11:30:28
```

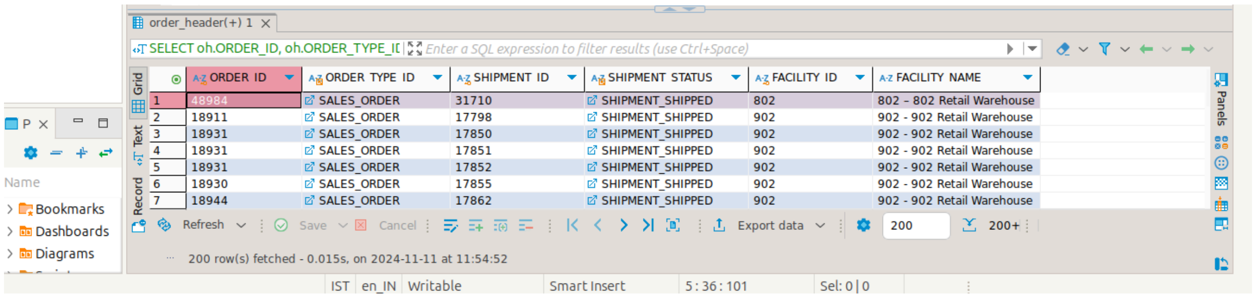
16. Send sale orders shipped from the warehouse:

Query:-

```sql
SELECT
  oh.ORDER_ID,
  oh.ORDER_TYPE_ID,
  s.SHIPMENT_ID,
  s.STATUS_ID AS SHIPMENT_STATUS,
  f.FACILITY_ID,
  f.FACILITY_NAME
FROM
  order_header oh
JOIN
  shipment s ON oh.ORDER_ID = s.PRIMARY_ORDER_ID
JOIN
  facility f ON s.ORIGIN_FACILITY_ID = f.FACILITY_ID
WHERE
  oh.ORDER_TYPE_ID = 'sales_order'
  AND f.FACILITY_TYPE_ID = 'warehouse'
  AND s.STATUS_ID = 'SHIPMENT_SHIPPED';
```

Explanation:-

This SQL query retrieves details of **sales orders** that have been **shipped from a warehouse**. It selects the **order ID**, **order type**, **shipment ID**, **shipment status**, and **facility details** (ID and name). The query joins the order_header table (which contains order details) with the shipment table (to link shipments to the orders) using the PRIMARY_ORDER_ID, and it also joins the facility table (to get warehouse details) using the ORIGIN_FACILITY_ID. The query filters for orders of type **'sales_order'**, ensures the warehouse facility type (FACILITY_TYPE_ID = 'warehouse'), and selects only shipments with a status of **'SHIPMENT_SHIPPED'**. This results in a list of sales orders that have been shipped from a warehouse.



```
SELECT oh.ORDER_ID, oh.ORDER_TYPE_I[
```

	ORDER ID	ORDER TYPE ID	SHIPMENT ID	SHIPMENT STATUS	FACILITY ID	FACILITY NAME
1	48984	SALES_ORDER	31710	SHIPMENT_SHIPPED	802	802 - 802 Retail Warehouse
2	18911	SALES_ORDER	17798	SHIPMENT_SHIPPED	902	902 - 902 Retail Warehouse
3	18931	SALES_ORDER	17850	SHIPMENT_SHIPPED	902	902 - 902 Retail Warehouse
4	18931	SALES_ORDER	17851	SHIPMENT_SHIPPED	902	902 - 902 Retail Warehouse
5	18931	SALES_ORDER	17852	SHIPMENT_SHIPPED	902	902 - 902 Retail Warehouse
6	18930	SALES_ORDER	17855	SHIPMENT_SHIPPED	902	902 - 902 Retail Warehouse
7	18944	SALES_ORDER	17862	SHIPMENT_SHIPPED	902	902 - 902 Retail Warehouse

```
200 row(s) fetched - 0.015s, on 2024-11-11 at 11:54:52
```

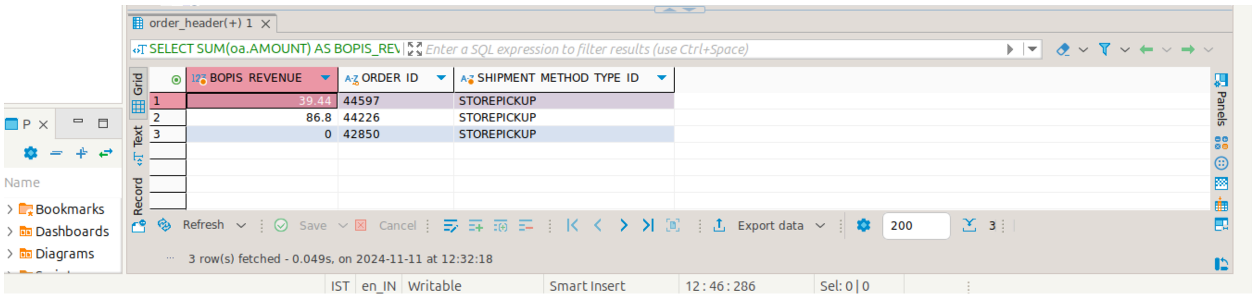
17. BOPIS orders Revenue in the last year:

Query:-

```
SELECT
  SUM(oa.AMOUNT) AS BOPIS_REVENUE,
  oh.ORDER_ID,
  s.SHIPMENT_METHOD_TYPE_ID
FROM
  order_header oh
JOIN
  order_adjustment oa ON oh.ORDER_ID = oa.ORDER_ID
JOIN
  shipment s ON oh.ORDER_ID = s.PRIMARY_ORDER_ID
WHERE
  s.SHIPMENT_METHOD_TYPE_ID = 'STOREPICKUP'
  AND s.CREATED_DATE >= NOW() - INTERVAL 1 YEAR
GROUP BY
  oh.ORDER_ID, s.SHIPMENT_METHOD_TYPE_ID;
```

Explanation:-

This SQL query calculates the total revenue from **BOPIS (Buy Online, Pickup In-Store)** orders that use the STOREPICKUP shipment method within the last year. It joins three tables: order_header, order_adjustment, and shipment, using ORDER_ID to link the order details, adjustments, and shipment information. The query filters the results to include only orders where the shipment method is STOREPICKUP and the CREATED_DATE of the shipment is within the past year. It then groups the data by ORDER_ID and SHIPMENT_METHOD_TYPE_ID to calculate the total revenue for each order based on the AMOUNT from the order_adjustment table. The result shows the total revenue per order and its associated shipment method.



order_header(+) 1

SELECT SUM(oa.AMOUNT) AS BOPIS_REV

	BOPIS_REVENUE	ORDER_ID	SHIPMENT_METHOD_TYPE_ID
1	39.44	44597	STOREPICKUP
2	86.8	44226	STOREPICKUP
3	0	42850	STOREPICKUP

3 row(s) fetched - 0.049s, on 2024-11-11 at 12:32:18