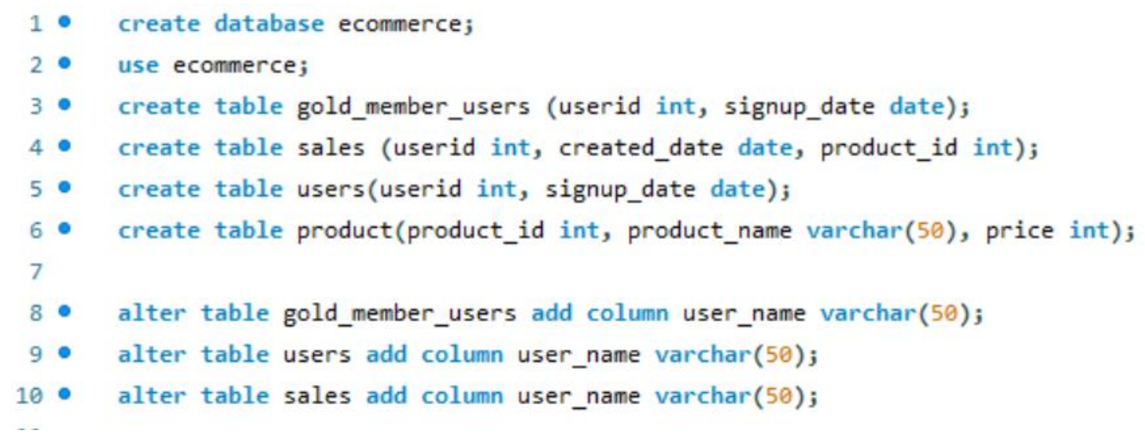
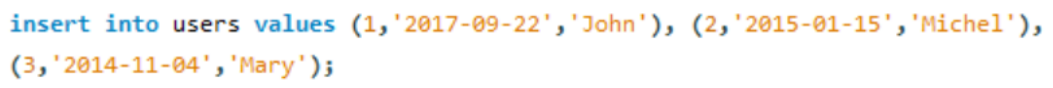
Question – 1

Database ecommerce created

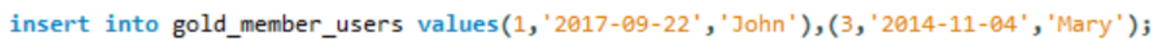


```
create database ecommerce;
use ecommerce;
create table gold_member_users (userid int, signup_date date);
create table sales (userid int, created_date date, product_id int);
create table users(userid int, signup_date date);
create table product(product_id int, product_name varchar(50), price int);

alter table gold_member_users add column user_name varchar(50);
alter table users add column user_name varchar(50);
alter table sales add column user_name varchar(50);
```

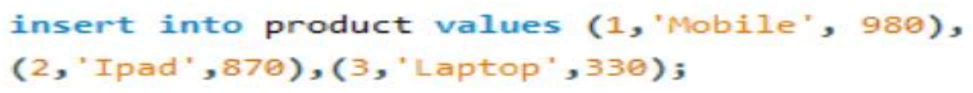
Data insert



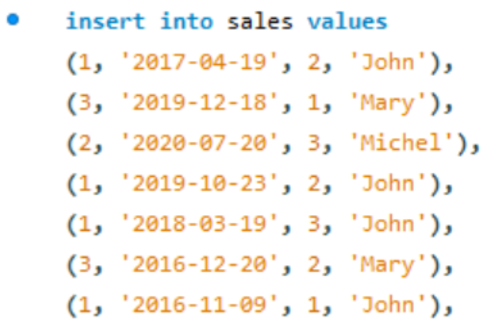
```
insert into users values (1,'2017-09-22','John'), (2,'2015-01-15','Michel'),
(3,'2014-11-04','Mary');
```



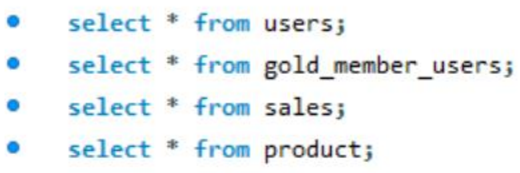
```
insert into gold_member_users values(1,'2017-09-22','John'),(3,'2014-11-04','Mary');
```



```
insert into product values (1,'Mobile', 980),
(2,'Ipad',870),(3,'Laptop',330);
```

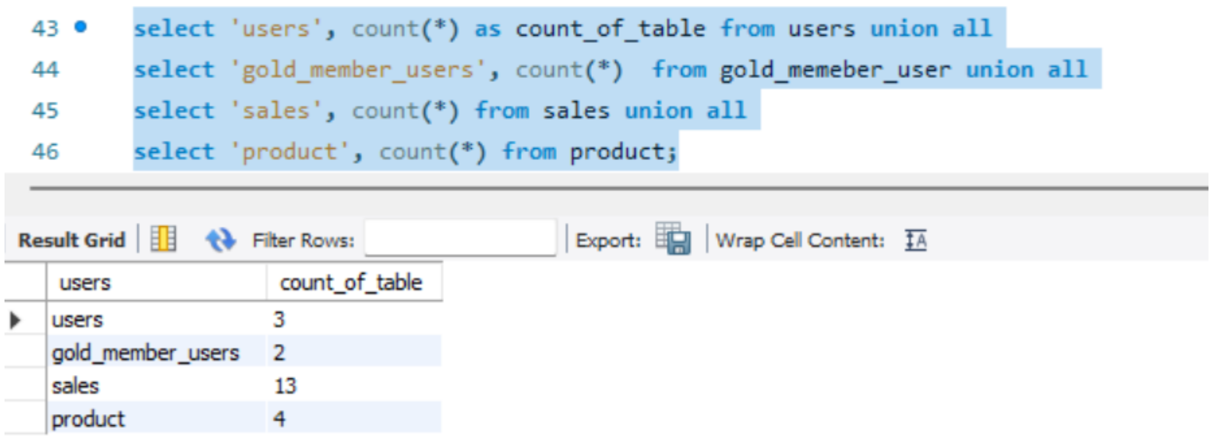


```
insert into sales values
(1, '2017-04-19', 2, 'John'),
(3, '2019-12-18', 1, 'Mary'),
(2, '2020-07-20', 3, 'Michel'),
(1, '2019-10-23', 2, 'John'),
(1, '2018-03-19', 3, 'John'),
(3, '2016-12-20', 2, 'Mary'),
(1, '2016-11-09', 1, 'John'),
```



```
select * from users;
select * from gold_member_users;
select * from sales;
select * from product;
```

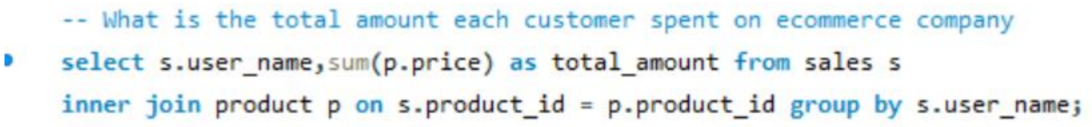
5. Count all the records of all four tables using single query



```sql
select 'users', count(*) as count_of_table from users union all
select 'gold_member_users', count(*)  from gold_memeber_user union all
select 'sales', count(*) from sales union all
select 'product', count(*) from product;
```

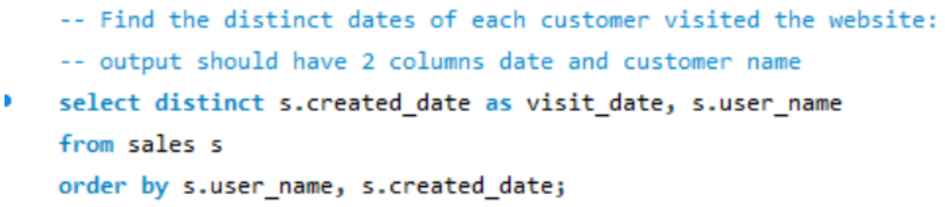
users	count_of_table
users	3
gold_member_users	2
sales	13
product	4

6. What is the total amount each customer spent on an ecommerce company



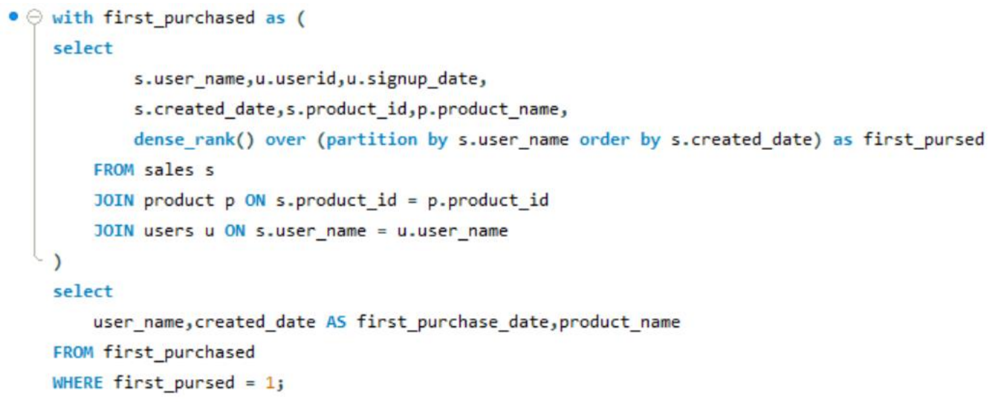
```sql
-- What is the total amount each customer spent on ecommerce company
select s.user_name,sum(p.price) as total_amount from sales s
inner join product p on s.product_id = p.product_id group by s.user_name;
```

7. Find the distinct dates of each customer visited the website: output should have 2 columns date and customer name



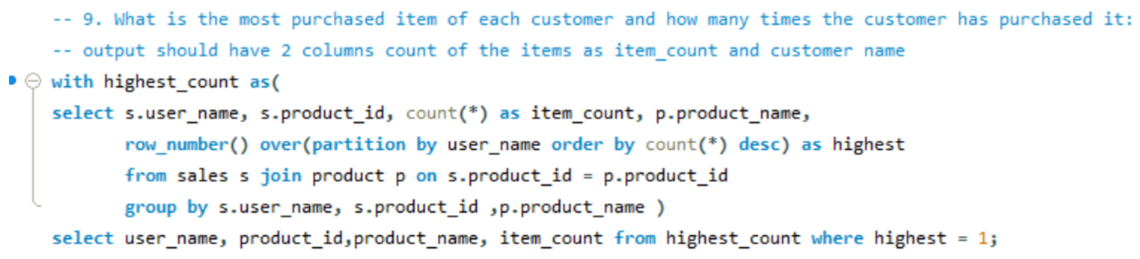
```sql
-- Find the distinct dates of each customer visited the website:
-- output should have 2 columns date and customer name
select distinct s.created_date as visit_date, s.user_name
from sales s
order by s.user_name, s.created_date;
```

8. Find the first product purchased by each customer using 3 tables (users, sales, product)



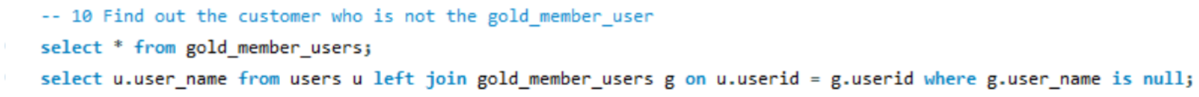
```sql
with first_purchased as (
select
        s.user_name,u.userid,u.signup_date,
        s.created_date,s.product_id,p.product_name,
        dense_rank() over (partition by s.user_name order by s.created_date) as first_pursed
    FROM sales s
    JOIN product p ON s.product_id = p.product_id
    JOIN users u ON s.user_name = u.user_name
)
select
    user_name,created_date AS first_purchase_date,product_name
FROM first_purchased
WHERE first_pursed = 1;
```

9. What is the most purchased item of each customer and how many times the customer has purchased it: output should have 2 columns count of the items as item_count and customer name



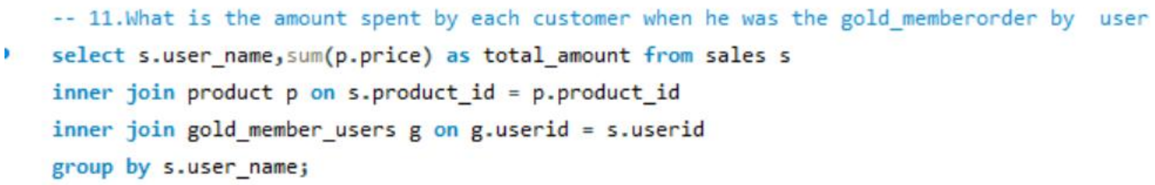
```
-- 9. What is the most purchased item of each customer and how many times the customer has purchased it:
-- output should have 2 columns count of the items as item_count and customer name
with highest_count as(
select s.user_name, s.product_id, count(*) as item_count, p.product_name,
       row_number() over(partition by user_name order by count(*) desc) as highest
       from sales s join product p on s.product_id = p.product_id
       group by s.user_name, s.product_id ,p.product_name )
select user_name, product_id,product_name, item_count from highest_count where highest = 1;
```

10. Find out the customer who is not the gold_member_user



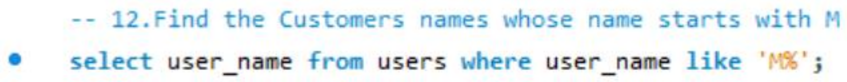
```
-- 10 Find out the customer who is not the gold_member_user
select * from gold_member_users;
select u.user_name from users u left join gold_member_users g on u.userid = g.userid where g.user_name is null;
```

11. What is the amount spent by each customer when he was the gold_memberorder by user



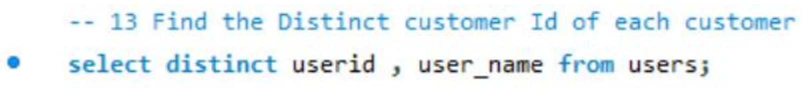
```
-- 11.What is the amount spent by each customer when he was the gold_memberorder by  user
select s.user_name,sum(p.price) as total_amount from sales s
inner join product p on s.product_id = p.product_id
inner join gold_member_users g on g.userid = s.userid
group by s.user_name;
```

12.Find the Customers names whose name starts with M



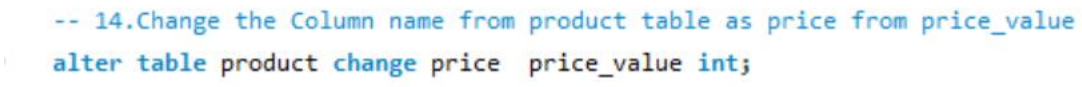
```
-- 12.Find the Customers names whose name starts with M
select user_name from users where user_name like 'M%';
```

13.Find the Distinct customer Id of each customer



```
-- 13 Find the Distinct customer Id of each customer
select distinct userid , user_name from users;
```

14.Change the Column name from product table as price_value from price



```
-- 14.Change the Column name from product table as price from price_value
alter table product change price  price_value int;
```

15.Change the Column value product_name – **Ipad to Iphone** from product table

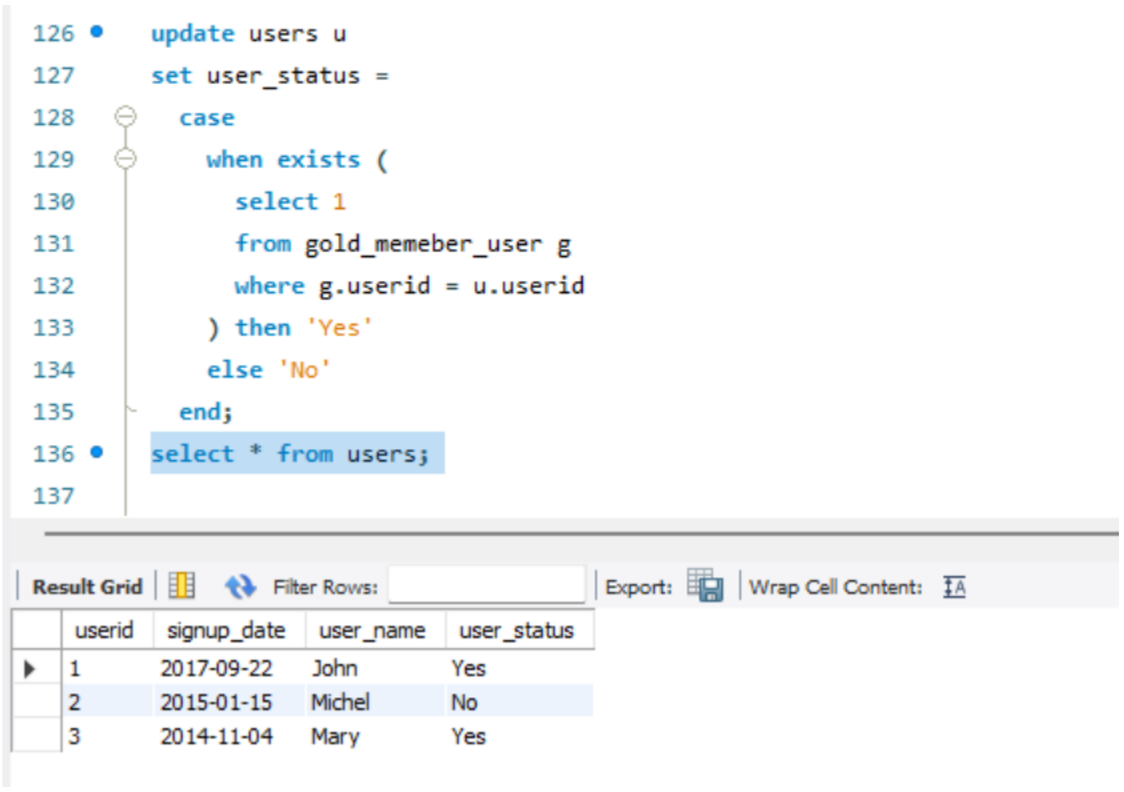
```
-- 15.Change the Column value product_name - Ipad to Iphone from product table
SET SQL_SAFE_UPDATES = 0;
update  product set product_name = "Iphone" where product_name = "Ipad";
select * from product;
```

16.Change the table name of **gold_member_users to gold_membership_users**

```
-- 16.Change the table name of gold_member_users to gold_membership_users
-- EXEC sp_rename 'old_table_name', 'new_table_name';
rename table gold_member_users to gold_memeber_user;
```

17.Create a new column as **Status** in the table crate above **gold_membership_users** the Status values should be 2 Yes and No if the user is gold member, then status should be Yes else No.

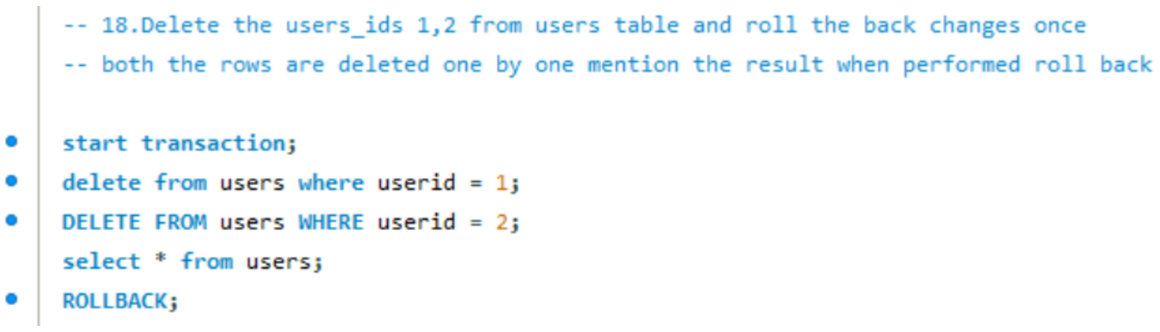
```
-- 17.Create a new column as Status in the table gold_membership_user
-- the Status values should be 2 Yes and No if the user is gold member, then status should be Ye
alter table gold_memeber_user add column status varchar(5);
alter table users add column user_status varchar(5);
```



```
update users u
set user_status =
  case
    when exists (
      select 1
      from gold_memeber_user g
      where g.userid = u.userid
    ) then 'Yes'
    else 'No'
  end;
select * from users;
```

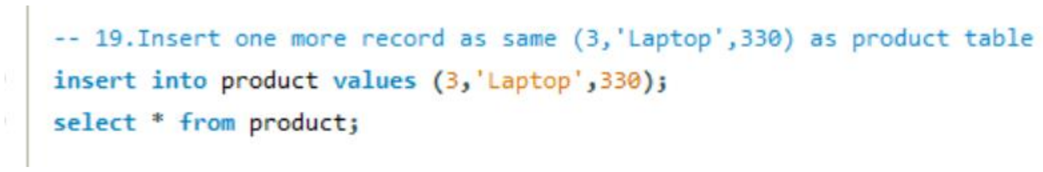
userid	signup_date	user_name	user_status
1	2017-09-22	John	Yes
2	2015-01-15	Michel	No
3	2014-11-04	Mary	Yes

18.Delete the users_ids 1,2 from users table and roll the back changes once both the rows are deleted one by one mention the result when performed roll back



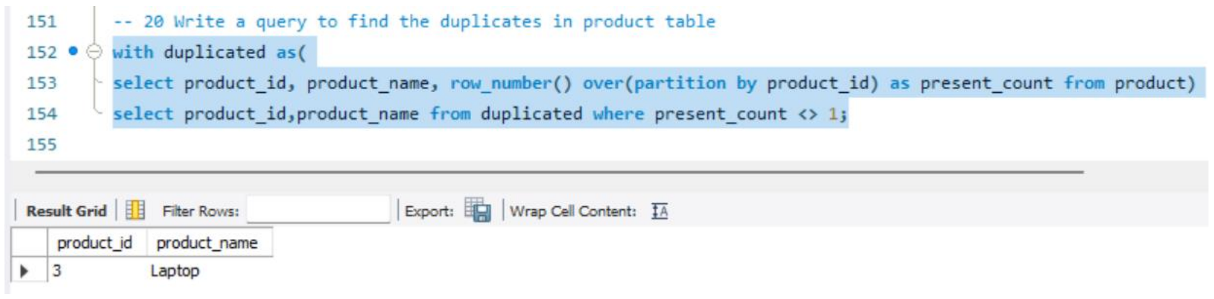
```
-- 18.Delete the users_ids 1,2 from users table and roll the back changes once
-- both the rows are deleted one by one mention the result when performed roll back

start transaction;
delete from users where userid = 1;
DELETE FROM users WHERE userid = 2;
select * from users;
ROLLBACK;
```

19.Insert one more record as same (3,'Laptop',330) as product table



```
-- 19.Insert one more record as same (3,'Laptop',330) as product table
insert into product values (3,'Laptop',330);
select * from product;
```

20.Write a query to find the duplicates in product table



```
-- 20 Write a query to find the duplicates in product table
with duplicated as(
select product_id, product_name, row_number() over(partition by product_id) as present_count from product)
select product_id,product_name from duplicated where present_count <> 1;
```

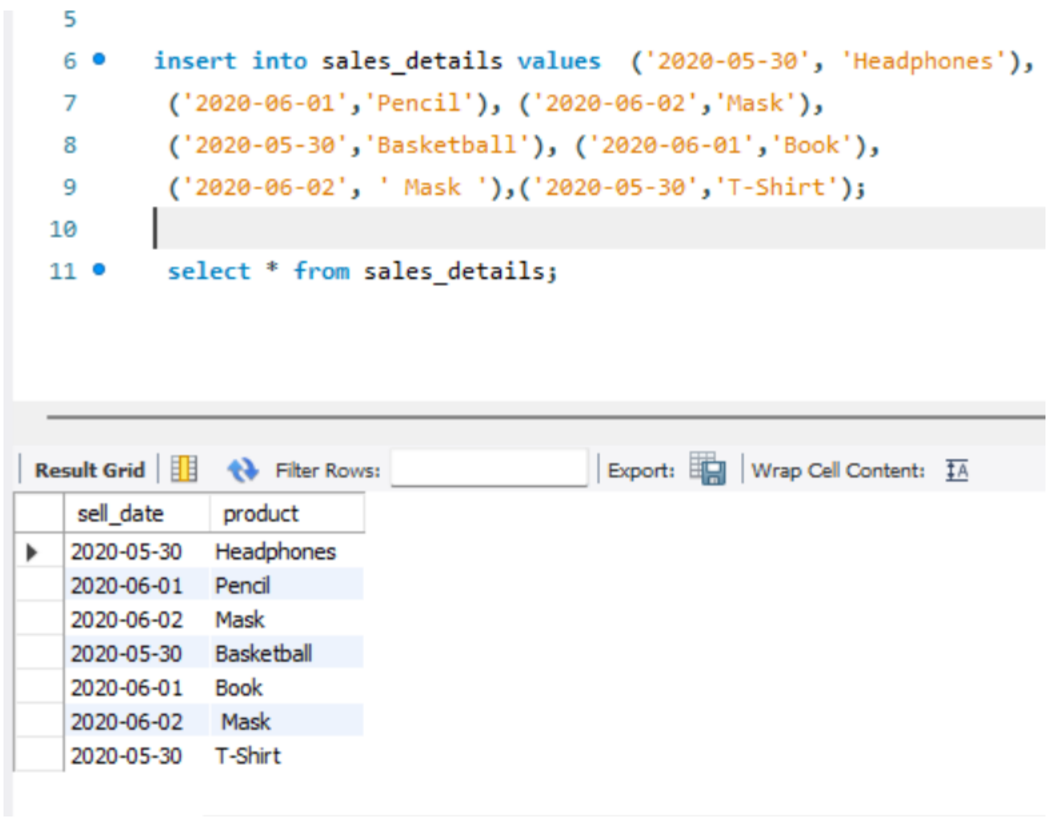
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
product_id   product_name
3            Laptop
```

Question - 2

Write a query to find for each date the number of different products sold and their names.

-- Column names: (sell_date, product)

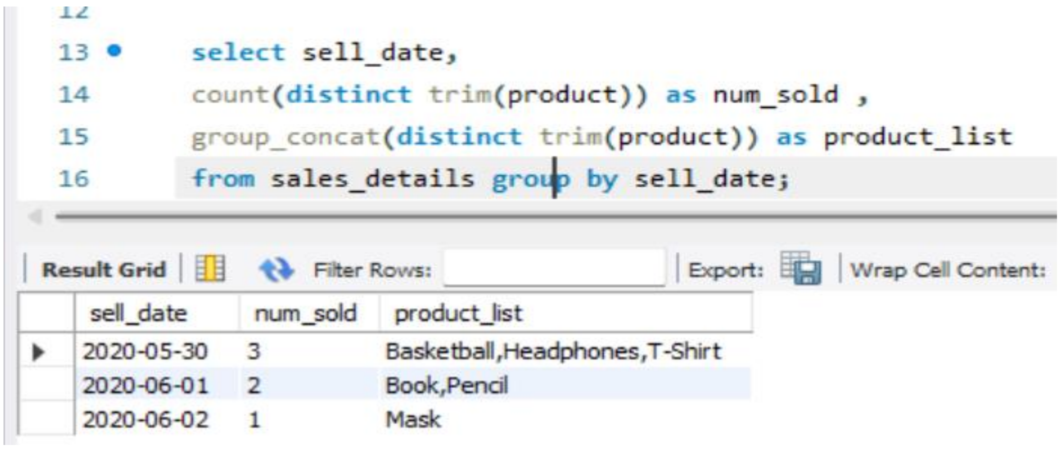


```sql
insert into sales_details values  ('2020-05-30', 'Headphones'),
 ('2020-06-01','Pencil'), ('2020-06-02','Mask'),
 ('2020-05-30','Basketball'), ('2020-06-01','Book'),
 ('2020-06-02', ' Mask '),('2020-05-30','T-Shirt');

 select * from sales_details;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sell_date	product
2020-05-30	Headphones
2020-06-01	Pencil
2020-06-02	Mask
2020-05-30	Basketball
2020-06-01	Book
2020-06-02	Mask
2020-05-30	T-Shirt

Output



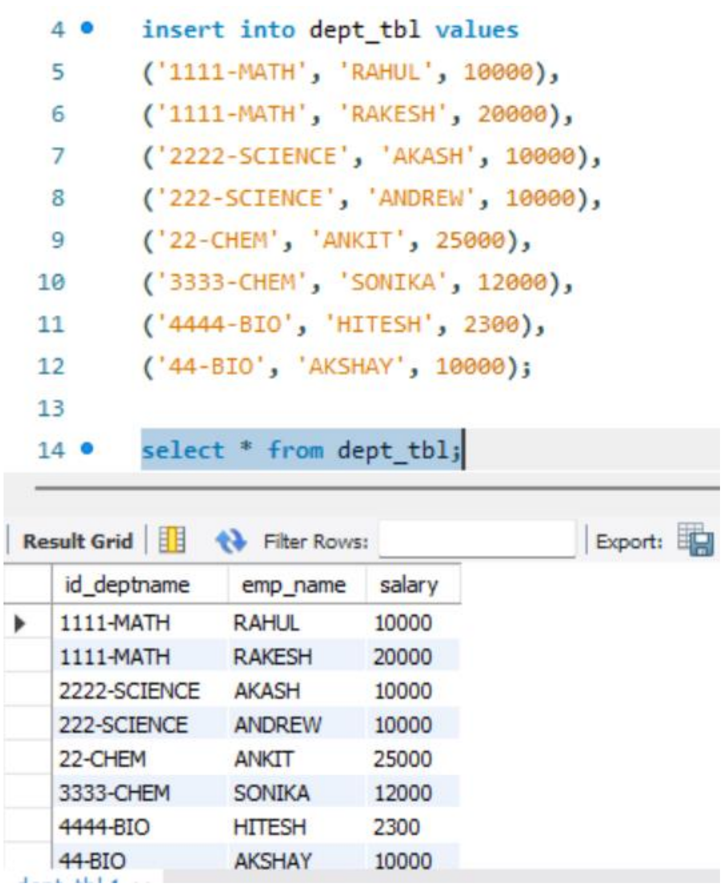
```sql
select sell_date,
count(distinct trim(product)) as num_sold ,
group_concat(distinct trim(product)) as product_list
from sales_details group by sell_date;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sell_date	num_sold	product_list
2020-05-30	3	Basketball,Headphones,T-Shirt
2020-06-01	2	Book,Pencil
2020-06-02	1	Mask

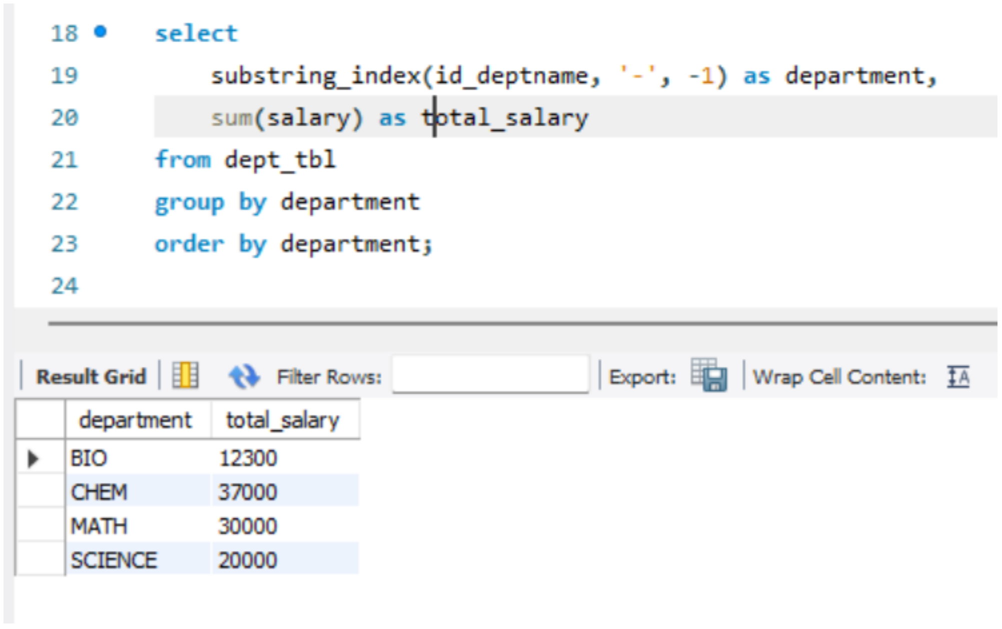
Question – 3

Find the total salary of each department



```sql
insert into dept_tbl values
('1111-MATH', 'RAHUL', 10000),
('1111-MATH', 'RAKESH', 20000),
('2222-SCIENCE', 'AKASH', 10000),
('222-SCIENCE', 'ANDREW', 10000),
('22-CHEM', 'ANKIT', 25000),
('3333-CHEM', 'SONIKA', 12000),
('4444-BIO', 'HITESH', 2300),
('44-BIO', 'AKSHAY', 10000);

select * from dept_tbl;
```

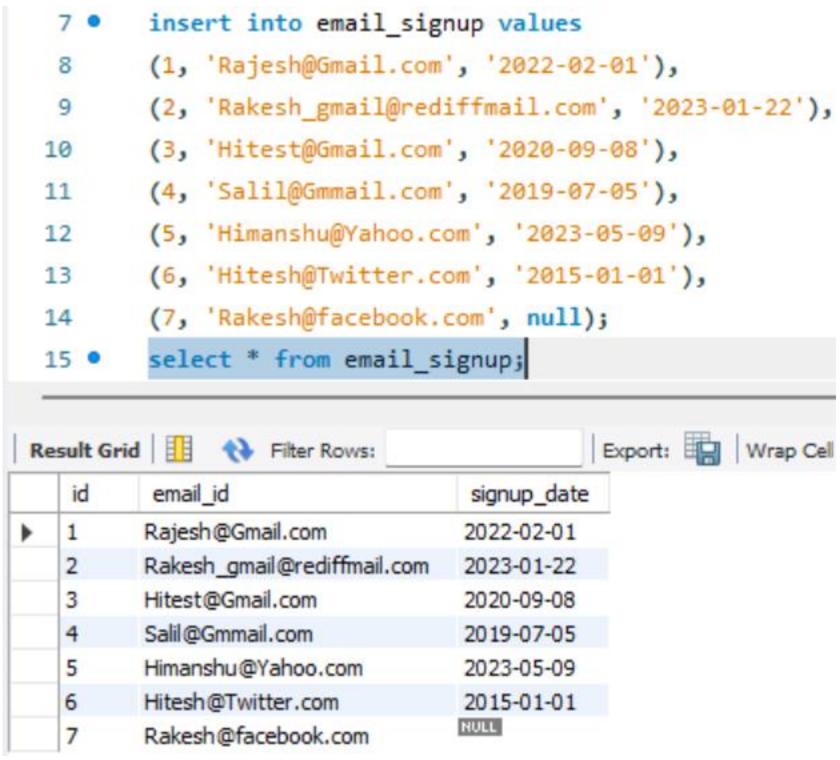
id_deptname	emp_name	salary
1111-MATH	RAHUL	10000
1111-MATH	RAKESH	20000
2222-SCIENCE	AKASH	10000
222-SCIENCE	ANDREW	10000
22-CHEM	ANKIT	25000
3333-CHEM	SONIKA	12000
4444-BIO	HITESH	2300
44-BIO	AKSHAY	10000



```sql
select
    substring_index(id_deptname, '-', -1) as department,
    sum(salary) as total_salary
from dept_tbl
group by department
order by department;
```

department	total_salary
BIO	12300
CHEM	37000
MATH	30000
SCIENCE	20000

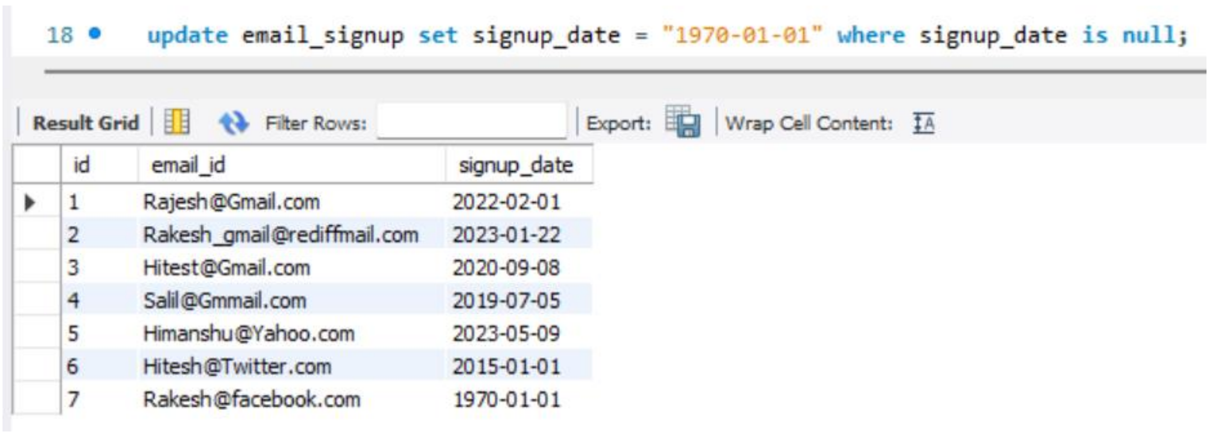
Question – 4



```sql
insert into email_signup values
(1, 'Rajesh@Gmail.com', '2022-02-01'),
(2, 'Rakesh_gmail@rediffmail.com', '2023-01-22'),
(3, 'Hitest@Gmail.com', '2020-09-08'),
(4, 'Salil@Gmmail.com', '2019-07-05'),
(5, 'Himanshu@Yahoo.com', '2023-05-09'),
(6, 'Hitesh@Twitter.com', '2015-01-01'),
(7, 'Rakesh@facebook.com', null);
select * from email_signup;
```

id	email_id	signup_date
1	Rajesh@Gmail.com	2022-02-01
2	Rakesh_gmail@rediffmail.com	2023-01-22
3	Hitest@Gmail.com	2020-09-08
4	Salil@Gmmail.com	2019-07-05
5	Himanshu@Yahoo.com	2023-05-09
6	Hitesh@Twitter.com	2015-01-01
7	Rakesh@facebook.com	NULL

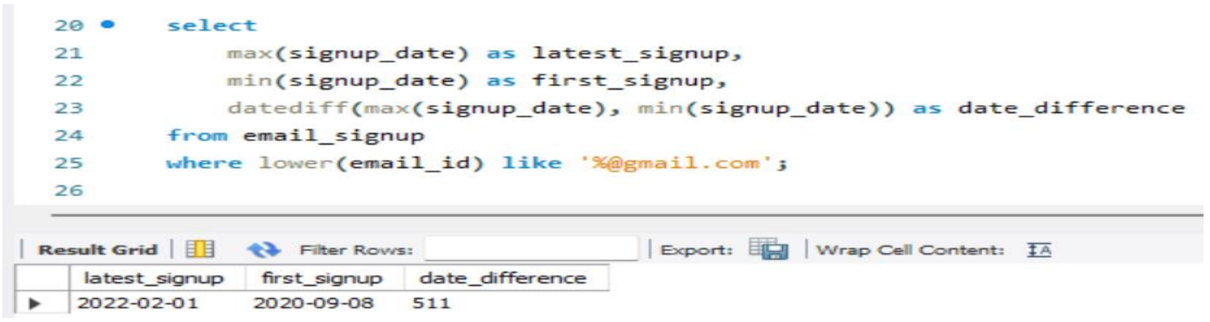
write the query to replace null value with '1970-01-01'



```sql
update email_signup set signup_date = "1970-01-01" where signup_date is null;
```

id	email_id	signup_date
1	Rajesh@Gmail.com	2022-02-01
2	Rakesh_gmail@rediffmail.com	2023-01-22
3	Hitest@Gmail.com	2020-09-08
4	Salil@Gmmail.com	2019-07-05
5	Himanshu@Yahoo.com	2023-05-09
6	Hitesh@Twitter.com	2015-01-01
7	Rakesh@facebook.com	1970-01-01

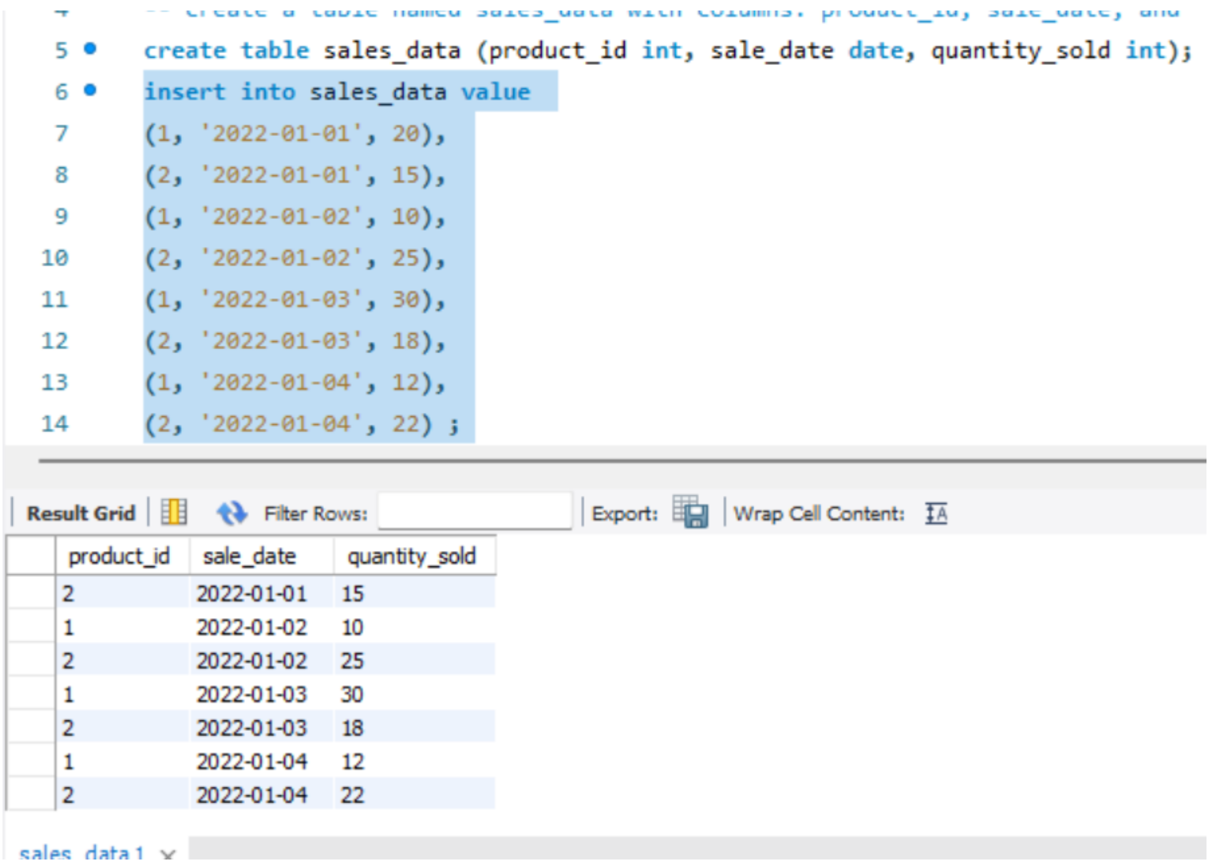
Date difference



```sql
select
    max(signup_date) as latest_signup,
    min(signup_date) as first_signup,
    datediff(max(signup_date), min(signup_date)) as date_difference
from email_signup
where lower(email_id) like '%@gmail.com';
```

latest_signup	first_signup	date_difference
2022-02-01	2020-09-08	511

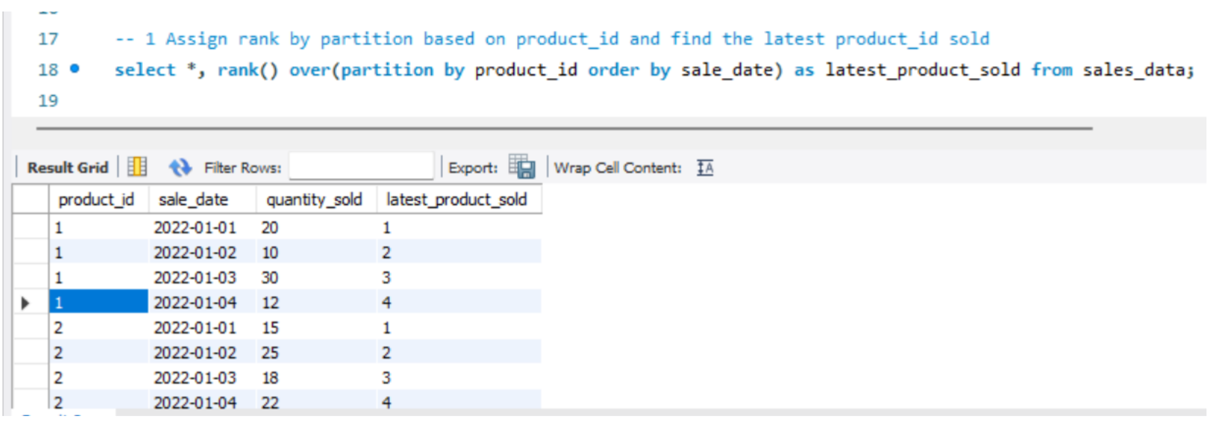
Question – 5



```sql
-- create a table named sales_data with columns: product_id, sale_date, and
create table sales_data (product_id int, sale_date date, quantity_sold int);
insert into sales_data value
(1, '2022-01-01', 20),
(2, '2022-01-01', 15),
(1, '2022-01-02', 10),
(2, '2022-01-02', 25),
(1, '2022-01-03', 30),
(2, '2022-01-03', 18),
(1, '2022-01-04', 12),
(2, '2022-01-04', 22) ;
```

product_id	sale_date	quantity_sold
2	2022-01-01	15
1	2022-01-02	10
2	2022-01-02	25
1	2022-01-03	30
2	2022-01-03	18
1	2022-01-04	12
2	2022-01-04	22

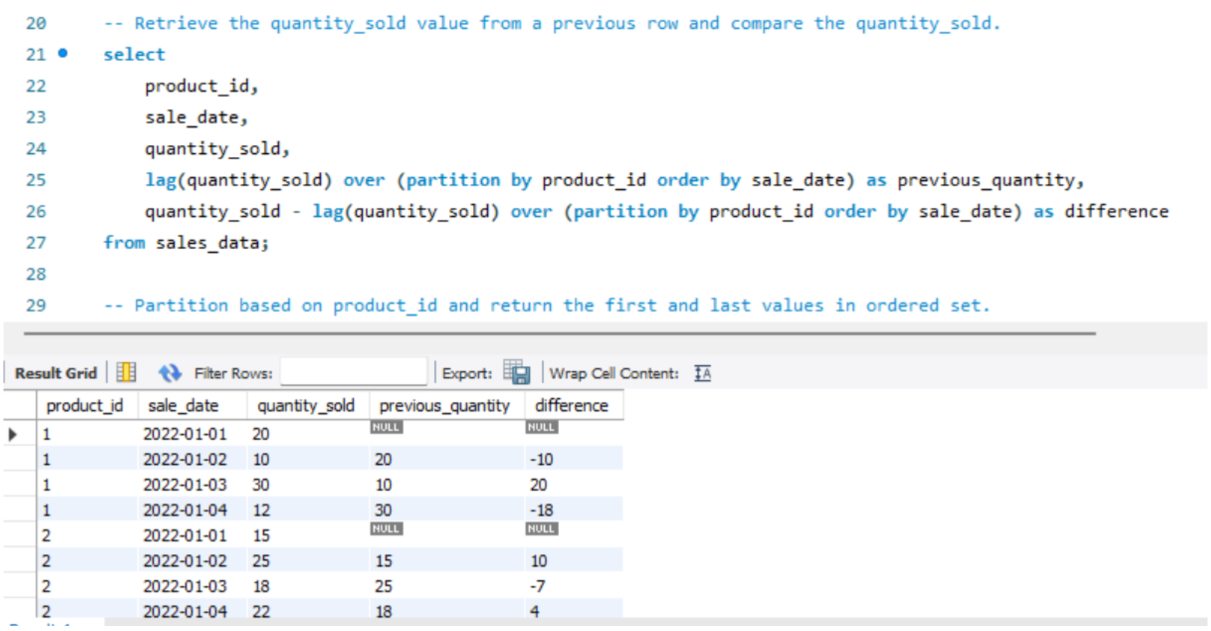
1) Assign rank by partition based on product_id and find the latest product_id sold



```sql
-- 1 Assign rank by partition based on product_id and find the latest product_id sold
select *, rank() over(partition by product_id order by sale_date) as latest_product_sold from sales_data;
```

product_id	sale_date	quantity_sold	latest_product_sold
1	2022-01-01	20	1
1	2022-01-02	10	2
1	2022-01-03	30	3
1	2022-01-04	12	4
2	2022-01-01	15	1
2	2022-01-02	25	2
2	2022-01-03	18	3
2	2022-01-04	22	4

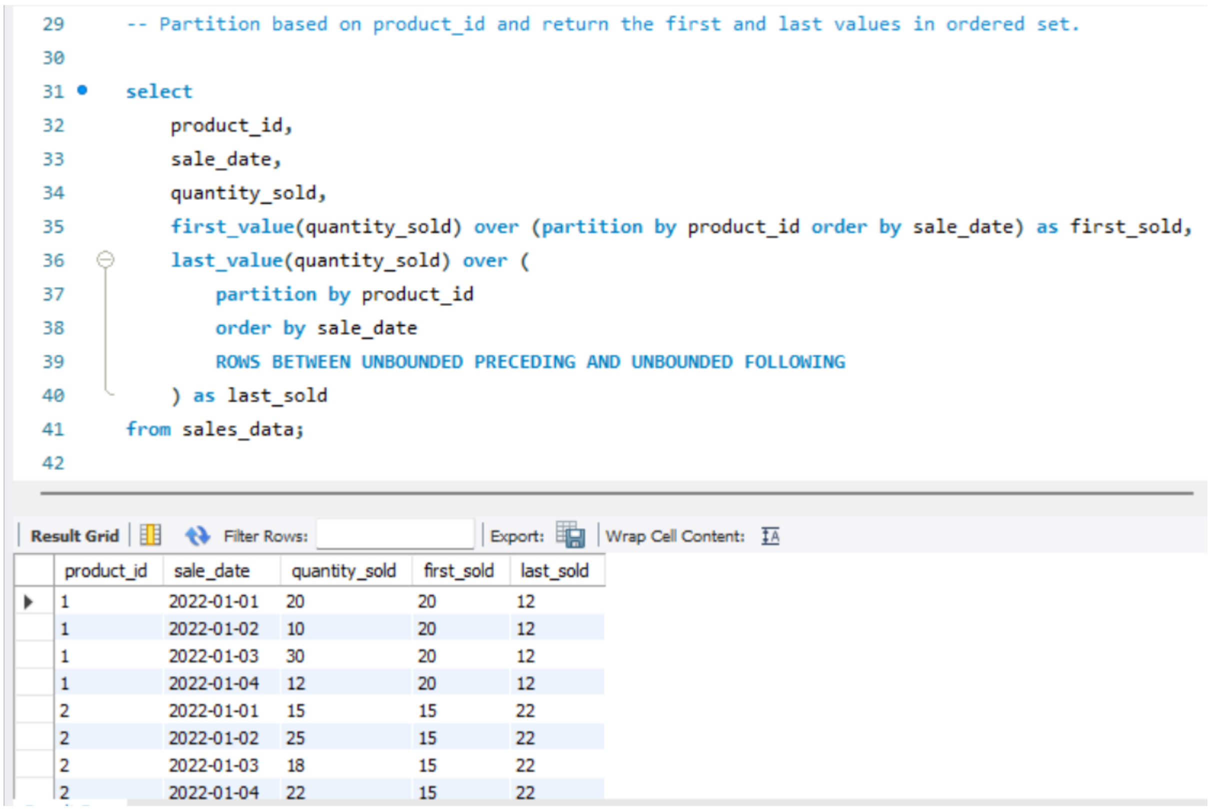
2) Retrieve the quantity_sold value from a previous row and compare the quantity_sold.



```sql
-- Retrieve the quantity_sold value from a previous row and compare the quantity_sold.
select
    product_id,
    sale_date,
    quantity_sold,
    lag(quantity_sold) over (partition by product_id order by sale_date) as previous_quantity,
    quantity_sold - lag(quantity_sold) over (partition by product_id order by sale_date) as difference
from sales_data;

-- Partition based on product_id and return the first and last values in ordered set.
```

product_id	sale_date	quantity_sold	previous_quantity	difference
1	2022-01-01	20	NULL	NULL
1	2022-01-02	10	20	-10
1	2022-01-03	30	10	20
1	2022-01-04	12	30	-18
2	2022-01-01	15	NULL	NULL
2	2022-01-02	25	15	10
2	2022-01-03	18	25	-7
2	2022-01-04	22	18	4

3) Partition based on product_id and return the first and last values in ordered set.



```sql
-- Partition based on product_id and return the first and last values in ordered set.

select
    product_id,
    sale_date,
    quantity_sold,
    first_value(quantity_sold) over (partition by product_id order by sale_date) as first_sold,
    last_value(quantity_sold) over (
        partition by product_id
        order by sale_date
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
    ) as last_sold
from sales_data;

```

product_id	sale_date	quantity_sold	first_sold	last_sold
1	2022-01-01	20	20	12
1	2022-01-02	10	20	12
1	2022-01-03	30	20	12
1	2022-01-04	12	20	12
2	2022-01-01	15	15	22
2	2022-01-02	25	15	22
2	2022-01-03	18	15	22
2	2022-01-04	22	15	22