SQL Assignment

1.Here is the Dataset for the below questions.

Gold members Data Set--Column names : (userid integer, signup_date)
('John','09-22-2017'), ('Mary','04-21-2017')
values
Users Data Set--Column names : (userid integer, signup_date)
('John','09-02-2014'), ('Michel','01-15-2015'), ('Mary','04-11-2014')

Sales Data Set-- Column names : (userid,created_date,product_id)
('John','04-19-2017',2), ('Mary','12-18-2019',1), ('Michel','07-20-2020',3), ('John','10-23-2019',2), ('John','03-19-2018',3), ('Mary','12-20-2016',2), ('John','11-09-2016',1), ('John','05-20-2016',3), ('Michel','09-24-2017',1), ('John','03-11-2017',2), ('John','03-11-2016',1), ('Mary','11-10-2016',1), ('Mary','12-07-2017',2)

Product Data Set--Column names : (product_id,product_name,price)
(1,'Mobile',980), (2,'Ipad',870), (3,'Laptop',330)

Questions on above Dataset :

1.Create Database as ecommerce

```
CREATE DATABASE ecommerce;
USE ecommerce;
```

2.Create 4 tables (gold_member_users, users, sales, product) under the above database(ecommerce)

```
create table gold_member_users(
user_id int primary key,
user_name varchar(20),
sign_up_date date
);

create table users(
user_id int primary key,
user_name varchar(20),
sign_up_date date
);


create table sales(
user_id int,
user_name varchar(20),
created_datre date,
product_id int,
foreign key (product_id) references product (product_id)
);

create table product(
product_id int primary key,
product_name varchar(20),
price int
);
```

3.Insert the values provided above with respective datatypes

```
insert into gold_member_users values
( 1,'John','09-22-2017'), ( 2,'Mary','04-21-2017')
;

insert into users values
(1,'John','09-02-2014'), (2,'Michel','01-15-2015'), (3,'Mary','04-11-2014')
;

insert into sales values
(1,'John','04-19-2017',2), (3,'Mary','12-18-2019',1), (2,'Michel','07-20-2020',3),
(1,'John','10-23-2019',2), (1,'John','03-19-2018',3), (3,'Mary','12-20-2016',2),
(1,'John','11-09-2016',1), (1,'John','05-20-2016',3), (2,'Michel','09-24-2017',1),
(1,'John','03-11-2017',2), (1,'John','03-11-2016',1), (3,'Mary','11-10-2016',1),
(3,'Mary','12-07-2017',2)
;

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

4.Show all the tables in the same database(ecommerce)

```
50
51      -- To show only table names in the current database:
52      SELECT name
53      FROM sys.tables;
54
```

59 % 7 0

Results Messages

	name
1	gold_member_users
2	users
3	product
4	sales
5	product_details
6	dept_tbl
7	email_signup
8	sales_data

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

```
65
66      --5. Count all the records of all four tables using single query
67      select count(*) from gold_member_users
68      union all
69      select count(*) from users
70      union all
71      select count(*) from sales
72      union all
73      select count(*) from product;
74
```

59 % 7 0

Results Messages

	(No column name)
1	2
2	3
3	13
4	4

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

```
--6. What is the total amount each customer spent on ecommerce company
select users.user_name,sum(price) as total_sum
from users
join sales on users.user_id = sales.user_id
join product on sales.product_id = product.product_id
group by users.user_name;
```

59 % 7 0

Results Messages

	user_name	total_sum
1	John	5230
2	Mary	3700
3	Michel	1310

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

```
--7. Find the distinct dates of each customer visited the website: output should have 2 columns date and customer name
select distinct created_datre,users.user_name
from users
join sales on users.user_id = sales.user_id
--order by user_name;
group by created_datre,users.user_name;
```

59 % 7 0

Results Messages

	user_name	total_sum
1	John	5230
2	Mary	3700
3	Michel	1310

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

```
--8. Find the first product purchased by each customer using 3 tables(users, sales, product)
select distinct u.user_name,s.created_datre,p.product_name
from(select *, row_number() over(partition by
user_id order by created_datre)
as row_num from sales)s
inner join users u on s.user_id=u.user_id
inner join product p on s.product_id = p.product_id
where s.row_num=1;
```

59 % 7 0

Results Messages

	user_name	created_datre	product_name
1	John	2016-03-11	Mobile
2	Mary	2016-11-10	Mobile
3	Michel	2017-09-24	Mobile

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

```
select count(*) as item_count,product.product_name, sales.user_name
from users
join sales on users.user_id=sales.user_id
join product on product.product_id=sales.product_id
group by sales.user_name,product.product_name
order by sales.user_name;
```

	item_count	product_name	user_name
1	3	Ipad	John
2	2	Laptop	John
3	2	Mobile	John
4	2	Ipad	Mary
5	2	Mobile	Mary
6	1	Laptop	Michel
7	1	Mobile	Michel

10.Find out the customer who is not the gold_member_user

```
108  --10. Find out the customer who is not the gold_member_user
109  select users.user_name
110  from gold_member_users g
111  right join users on g.user_name = users.user_name
112  where g.sign_up_date IS NULL;
113
```

59 % 7 0

Results Messages

	user_name
1	Michel

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

```
114  --11.What is the amount spent by each customer when he was the gold_member order by  user
115  SELECT g.user_id,g.user_name,SUM(p.price) AS total_amount_spent
116  FROM sales s
117  JOIN product p ON s.product_id = p.product_id
118  JOIN gold_member_users g ON s.user_id = g.user_id
119  WHERE s.created_datre >= g.sign_up_date
120  GROUP BY g.user_id, g.user_name
121  ORDER BY g.user_id;
122
```

59 % 7 0

Results Messages

	user_id	user_name	total_amount_spent
1	1	John	1200
2	2	Mary	1310

12.Find the Customers names whose name starts with M

```
123  --12.Find the Customers names whose name starts with M
124  SELECT user_name
125  FROM users
126  WHERE user_name LIKE 'M%';
127
```

59 % 7 0

Results Messages

	user_name
1	Michel
2	Mary

13.Find the Distinct customer Id of each customer

```
127
128  --13.Find the Distinct customer Id of each customer
129  SELECT DISTINCT user_id
130  FROM sales;
131
132  --14.Change the Column name from product table as price_valu
```

59 % 7 0

Results Messages

	user_id
1	1
2	2
3	3

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

```
132  --14.Change the Column name from product table as price_value from price
133  EXEC sp_rename 'product.price', 'price_value', 'COLUMN';
134  select * from product;
```

59 % 7 0

Results Messages

	product_id	product_name	price_value
1	1	Mobile	980
2	2	Ipad	870
3	3	Laptop	330
4	4	Laptop	330

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

```
136  --15.Change the Column value product_name - Ipad to Iphone from product table
137  UPDATE product
138  SET product_name = 'Iphone'
139  WHERE product_name = 'Ipad';
140
```

59 % 7 0

Results Messages

	product_id	product_name	price_value
1	1	Mobile	980
2	2	Iphone	870
3	3	Laptop	330
4	4	Laptop	330

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

```
141  --16.Change the table name of gold_member_users to gold_membership_users
142  EXEC sp_rename 'gold_member_users', 'gold_membership_users';
143
```

59 % 7 0

Results Messages

	name
1	gold_membership_users

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 crate above gold_membership_users  the Status val
--Step 1: Add the New Column to users Table
ALTER TABLE users
ADD Status VARCHAR(5);
--ALTER TABLE users
--Drop column Status ;
--Step 2: Update Status Based on Gold Membership
UPDATE users
SET Status = 'Yes'
WHERE users.user_name IN (
    SELECT gold_membership_users.user_name FROM gold_membership_users
);
--Step 3: Set Remaining Users' Status to 'No'
UPDATE users
SET Status = 'No'
WHERE Status IS NULL;
select * from users;
```

	user_id	user_name	sign_up_date	Status
1	1	John	2014-09-02	Yes
2	2	Michel	2015-01-15	No
3	3	Mary	2014-04-11	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 dele
BEGIN TRANSACTION;
-- Step 1: Delete user_id 1
DELETE FROM users WHERE user_id = 1;
-- Step 2: Delete user_id 2
DELETE FROM users WHERE user_id = 2;
-- Step 3: Check current state (these two users are now deleted temporarily)
SELECT * FROM users;
```

	user_id	user_name	sign_up_date	Status
1	3	Mary	2014-04-11	Yes

```
170  --Rollback
171  ROLLBACK;
172  SELECT * FROM users;
173
```

59 % 9 0

Results Messages

	user_id	user_name	sign_up_date	Status
1	1	John	2014-09-02	Yes
2	2	Michel	2015-01-15	No
3	3	Mary	2014-04-11	Yes

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

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

```
174  --19.Insert one more record as same (3,'Laptop',330) as product table
175  INSERT INTO product (product_id, product_name, price_value)
176  VALUES (5, 'Laptop', 330);
177
178  --20.Write a query to find the duplicates in product table
179  SELECT product_name, price_value, COUNT(*) AS duplicate_count
180  FROM product
181  GROUP BY product_name, price_value
182  HAVING COUNT(*) > 1;
```

59 % 12 0

Results Messages

	product_name	price_value	duplicate_count
1	Laptop	330	3

2. Write a query to find for each date the number of different products sold and their names.
Column names: (sell_date, product)

Data: ('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')

- Create table product_details with above data and find the above result
- Output:

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

```sql
CREATE TABLE product_details (
    sell_date DATE,
    product VARCHAR(50)
);

--drop table product_details

INSERT INTO product_details (sell_date, product)
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 product_details;
```

	sell_date	product
1	2020-05-30	Headphones
2	2020-06-01	Pencil
3	2020-06-02	Mask
4	2020-05-30	Basketball
5	2020-06-01	Book
6	2020-06-02	Mask
7	2020-05-30	T-Shirt

```sql
SELECT
    sell_date,
    COUNT(*) AS num_sold,
    STRING_AGG(LTRIM(RTRIM(product)), ', ')
       WITHIN GROUP (ORDER BY product) AS product_list
FROM product_details
GROUP BY sell_date
ORDER BY sell_date;
```

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

3. Find the total salary of each department

Column names:(id_deptname, emp_name, salary)

Data:
('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)

- Create table dept_tbl with above data

Output:

dept_name	total_salary
BIO	12300
CHEM	37000
Math	30000
Science	20000

```
CREATE TABLE dept_tbl (
    id_deptname VARCHAR(50),
    emp_name VARCHAR(50),
    salary INT
);

INSERT INTO dept_tbl (id_deptname, emp_name, salary)
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
1	1111-MATH	RAHUL	10000
2	1111-MATH	RAKESH	20000
3	2222-SCIENCE	AKASH	10000
4	222-SCIENCE	ANDREW	10000
5	22-CHEM	ANKIT	25000
6	3333-CHEM	SONIKA	12000
7	4444-BIO	HITESH	2300
8	44-BIO	AKSHAY	10000

```
select
RIGHT(id_deptname, LEN(id_deptname) - CHARINDEX('-', id_deptname)) as deptname,
sum(salary)
from dept_tbl
group by right(id_deptname, LEN(id_deptname) - CHARINDEX('-', id_deptname))
order by deptname;
```

	deptname	(No column name)
1	BIO	12300
2	CHEM	37000
3	MATH	30000
4	SCIENCE	20000

4.Write a query to find gmail accounts with latest and first signup date and difference between both the dates and also write the query to replace null value with '1970-01-01'

Column names: (id, email_id, signup_date)

Data:

(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);

Create table email_signup with above data

Output:

count_gmail_account	latest_signup_date	first_signup_date	diff_in_days
2	2022-02-01	2020-09-08	511

```sql
CREATE TABLE email_signup (
    id INT,
    email_id VARCHAR(100),
    signup_date DATE
);

INSERT INTO email_signup (id, email_id, signup_date)
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);
```

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

```sql
SELECT
    COUNT(*) AS count_gmail_account,
    MAX(signup_date) AS latest_signup_date,
    MIN(signup_date) AS first_signup_date,
    DATEDIFF(DAY, MIN(signup_date), MAX(signup_date)) AS diff_in_days
FROM email_signup
WHERE LOWER(email_id) LIKE '%@gmail.com';

UPDATE email_signup
SET signup_date = '1970-01-01'
WHERE signup_date IS NULL;
```

	count_gmail_account	latest_signup_date	first_signup_date	diff_in_days
1	2	2022-02-01	2020-09-08	511

5) Solve the below questions by creating below mentioned tables and adding the given dataset.

Hint: Solve using the Window Functions

1) create a table named **`sales_data`** with columns: **`product_id`**, **`sale_date`**, and **`quantity_sold`**.
2) insert some sample data into the table:
   dataset:

   (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)

```sql
CREATE TABLE sales_data (
    product_id INT,
    sale_date DATE,
    quantity_sold INT
);

INSERT INTO sales_data (product_id, sale_date, quantity_sold) VALUES
(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);

select * from sales_data;
```

	product_id	sale_date	quantity_sold
1	1	2022-01-01	20
2	2	2022-01-01	15
3	1	2022-01-02	10
4	2	2022-01-02	25
5	1	2022-01-03	30
6	2	2022-01-03	18
7	1	2022-01-04	12
8	2	2022-01-04	22

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

```sql
--Assign Rank by product_id and Find the Latest Sold Date per Product
SELECT *,
       RANK() OVER (PARTITION BY product_id ORDER BY sale_date DESC) AS rank_latest
FROM sales_data;
```

	product_id	sale_date	quantity_sold	rank_latest
1	1	2022-01-04	12	1
2	1	2022-01-03	30	2
3	1	2022-01-02	10	3
4	1	2022-01-01	20	4
5	2	2022-01-04	22	1
6	2	2022-01-03	18	2
7	2	2022-01-02	25	3
8	2	2022-01-01	15	4

4) Retrieve the **`quantity_sold`** value from a previous row and compare the **`quantity_sold`**.

```
--Retrieve quantity_sold from Previous Row and Compare
SELECT *,
       LAG(quantity_sold) OVER (PARTITION BY product_id ORDER BY sale_date) AS prev_qty_sold,
       quantity_sold - LAG(quantity_sold) OVER (PARTITION BY product_id ORDER BY sale_date) AS diff_qty
FROM sales_data;
```

	product_id	sale_date	quantity_sold	prev_qty_sold	diff_qty
1	1	2022-01-01	20	NULL	NULL
2	1	2022-01-02	10	20	-10
3	1	2022-01-03	30	10	20
4	1	2022-01-04	12	30	-18
5	2	2022-01-01	15	NULL	NULL
6	2	2022-01-02	25	15	10
7	2	2022-01-03	18	25	-7
8	2	2022-01-04	22	18	4

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

```
--Return First and Last Quantity Sold (per product_id partition)
SELECT *,
       FIRST_VALUE(quantity_sold) OVER (PARTITION BY product_id ORDER BY sale_date) AS first_qty_sold,
       LAST_VALUE(quantity_sold) OVER (PARTITION BY product_id ORDER BY sale_date
           ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS last_qty_sold
FROM sales_data;
```

	product_id	sale_date	quantity_sold	first_qty_sold	last_qty_sold
1	1	2022-01-01	20	20	12
2	1	2022-01-02	10	20	12
3	1	2022-01-03	30	20	12
4	1	2022-01-04	12	20	12
5	2	2022-01-01	15	15	22
6	2	2022-01-02	25	15	22
7	2	2022-01-03	18	15	22
8	2	2022-01-04	22	15	22