SE-COMP A BATCH-C	Roll number : 9913
Experiment no. : 5	Date of Implementation :27/2/2024

Aim : To implement simple SQL commands, string manipulation operations and aggregate functions.

Tool Used : PostgreSQL/Mysql

Related Course outcome : At the end of the course, Students will be able to Use SQL : Standard language of relational database

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness ● Maintains assignment deadline (3)	Assignment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness ● Complete all parts of QUERY assignment(3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality ● Extent of plagiarism(2)	Copied it from someone else(0)	At least few questions have been done without copying(1)	Assignment has been solved completely without copying (2)
Knowledge ● In depth knowledge of the QUERY assignment(2)	Unable to answer 2 questions(0)	Unable to answer 1 question (1)	Able to answer 2 questions (2)

Assessment Marks :

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Total : **(Out of 10)**

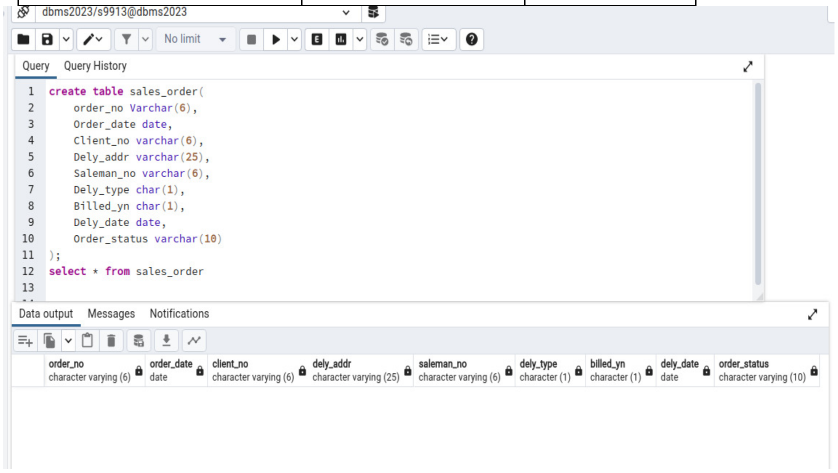
Teacher's Sign :

EXPERIMENT 5	Basic SQL Commands
Aim	To implement simple SQL commands, string manipulation operations and aggregate functions.
Tools	PostgreSQL
Theory	**SELECT:** SELECT statement returns a result set of records from one or more tables. The select statement has optional clauses: ● WHERE specifies which rows to retrieve ● GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group having group. ● HAVING selects among the groups defined by the GROUP BY clause. ● ORDER BY specifies an order in which to return the rows. Syntax: SELECT<attribute list> FROM<table list> WHERE<condition> Where ● Attribute list is a list of attribute name whose values to be retrieved by the query. ● Table list is a list of table name required to process query. ● Condition is a Boolean expression that identifies the tuples to be retrieved by query. **SQL Aggregate Functions** SQL aggregate functions return a single value, calculated from values in a column. Useful aggregate functions: ● AVG() - Returns the average value ● COUNT() - Returns the number of rows ● FIRST() - Returns the first value ● LAST() - Returns the last value ● MAX() - Returns the largest value ● MIN() - Returns the smallest value ● SUM() - Returns the sum **The SQL ORDER BY Keyword** The ORDER BY keyword is used to sort the result-set by one or more columns. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword. **SQL ORDER BY Syntax** SELECT column_name1, column_name2 FROM table_name ORDER BY column_name1 ASC|DESC, column_name2 ASC|DESC;

	Procedure	TASK 1:1. Create following table:

Table name : sales_order

Column Name	Data type	Size
order_no	varchar	6
Order_date	date	
Client_no	varchar	6
Dely_addr	varchar	25
Salesman_no	varchar	6
Dely_type	char	1
Billed_yn	char	1
Dely_date	Date	
Order_status	varchar	10


dbms2023/s9913@dbms2023

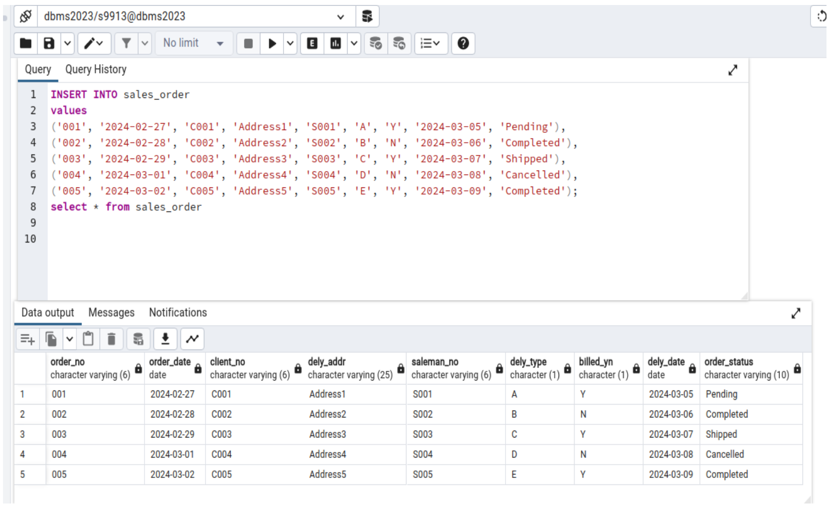
Query | Query History

```sql
create table sales_order(
    order_no Varchar(6),
    Order_date date,
    Client_no varchar(6),
    Dely_addr varchar(25),
    Saleman_no varchar(6),
    Dely_type char(1),
    Billed_yn char(1),
    Dely_date date,
    Order_status varchar(10)
);
select * from sales_order
```

Data output | Messages | Notifications

	order_no character varying (6)	order_date date	client_no character varying (6)	dely_addr character varying (25)	saleman_no character varying (6)	dely_type character (1)	billed_yn character (1)	dely_date date	order_status character varying (10)

2. Insert 5-6 records in table.


dbms2023/s9913@dbms2023

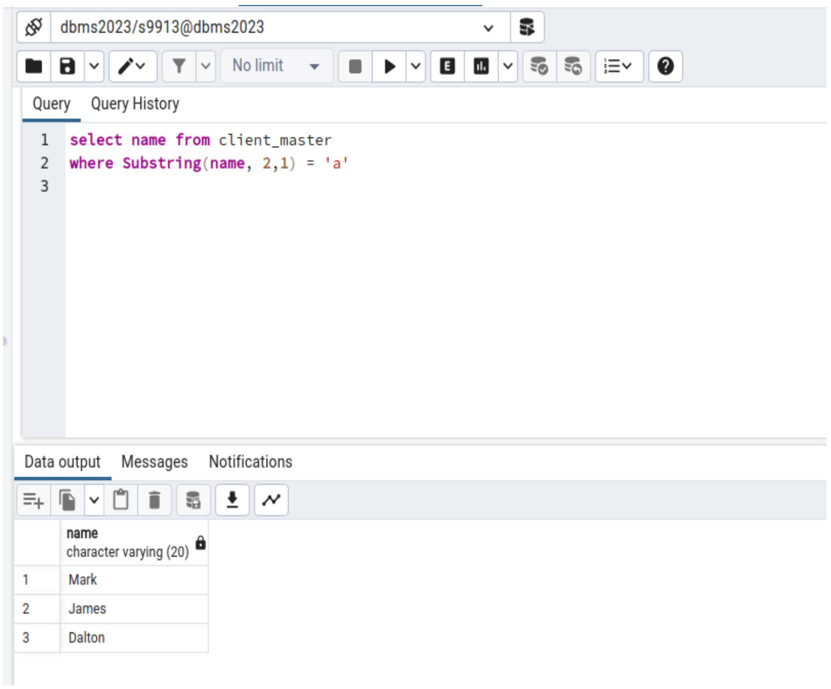
Query | Query History

```sql
INSERT INTO sales_order
values
('001', '2024-02-27', 'C001', 'Address1', 'S001', 'A', 'Y', '2024-03-05', 'Pending'),
('002', '2024-02-28', 'C002', 'Address2', 'S002', 'B', 'N', '2024-03-06', 'Completed'),
('003', '2024-02-29', 'C003', 'Address3', 'S003', 'C', 'Y', '2024-03-07', 'Shipped'),
('004', '2024-03-01', 'C004', 'Address4', 'S004', 'D', 'N', '2024-03-08', 'Cancelled'),
('005', '2024-03-02', 'C005', 'Address5', 'S005', 'E', 'Y', '2024-03-09', 'Completed');
select * from sales_order
```

Data output | Messages | Notifications

	order_no character varying (6)	order_date date	client_no character varying (6)	dely_addr character varying (25)	saleman_no character varying (6)	dely_type character (1)	billed_yn character (1)	dely_date date	order_status character varying (10)
1	001	2024-02-27	C001	Address1	S001	A	Y	2024-03-05	Pending
2	002	2024-02-28	C002	Address2	S002	B	N	2024-03-06	Completed
3	003	2024-02-29	C003	Address3	S003	C	Y	2024-03-07	Shipped
4	004	2024-03-01	C004	Address4	S004	D	N	2024-03-08	Cancelled
5	005	2024-03-02	C005	Address5	S005	E	Y	2024-03-09	Completed

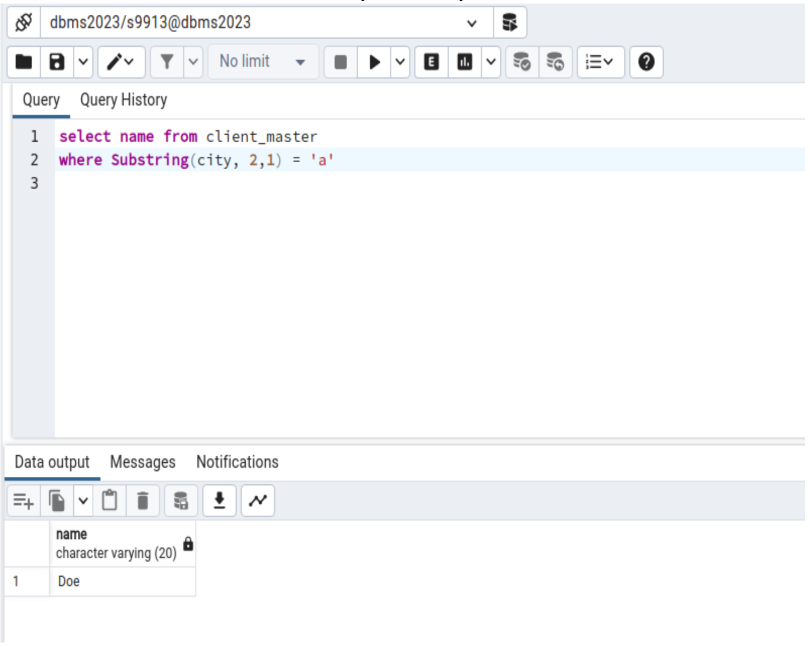
3. Find the names of all clients having ‘a’ as the second letter in their names.



```
select name from client_master
where Substring(name, 2,1) = 'a'
```

```
  name
  character varying (20)
1 Mark
2 James
3 Dalton
```

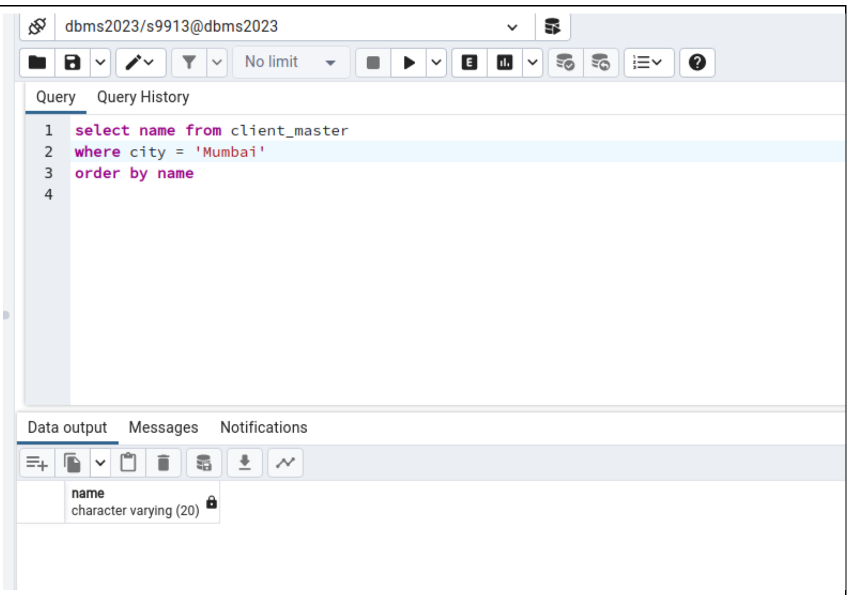
4. Find out the clients who stay in a city whose second letter is ‘a’



```
select name from client_master
where Substring(city, 2,1) = 'a'
```

```
  name
  character varying (20)
1 Doe
```

5. Find the list of all clients who stay in ‘mumbai’ ordered by their names


dbms2023/s9913@dbms2023

No limit

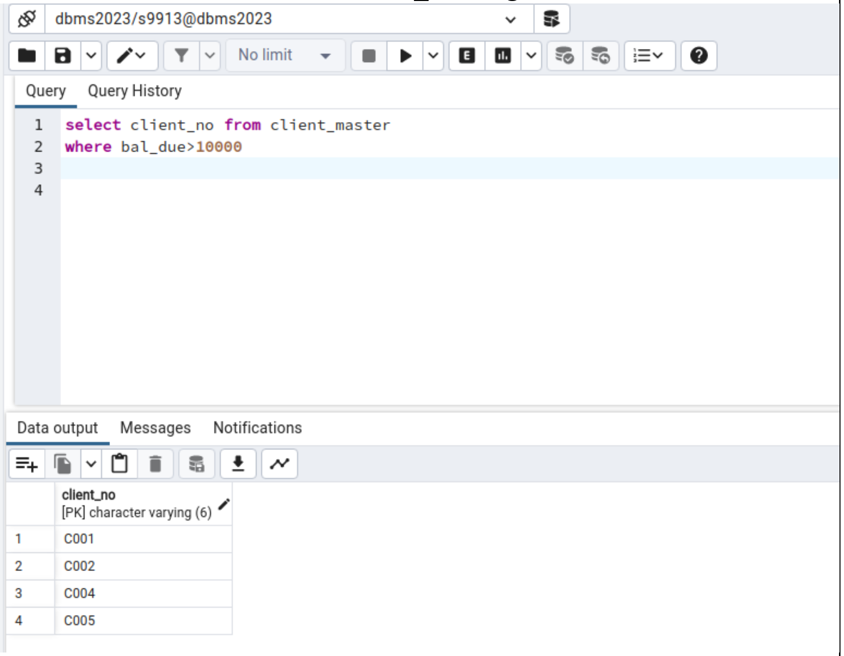
Query Query History

```sql
select name from client_master
where city = 'Mumbai'
order by name
```

Data output Messages Notifications

	name character varying (20)

6. Print the list of clients whose bal_due is greater than value 10000


dbms2023/s9913@dbms2023

No limit

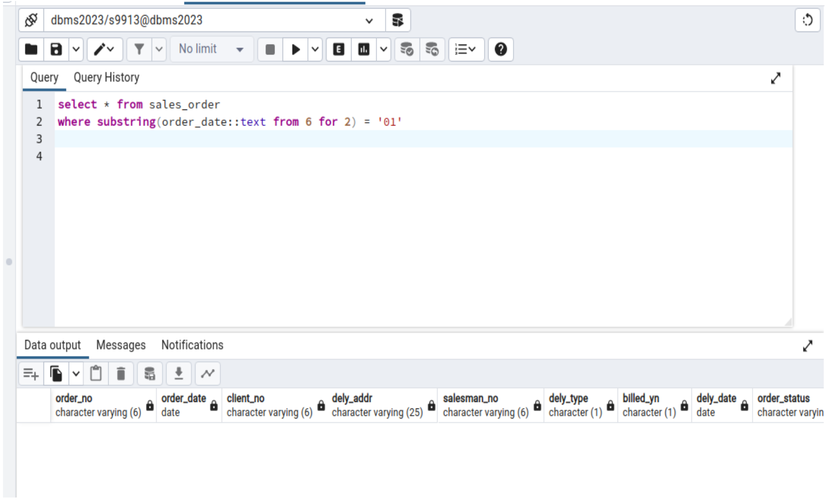
Query Query History

```sql
select client_no from client_master
where bal_due>10000
```

Data output Messages Notifications

	client_no [PK] character varying (6)
1	C001
2	C002
3	C004
4	C005

7. Print the information from sales_order table for orders placed in the month of January


dbms2023/s9913@dbms2023

No limit

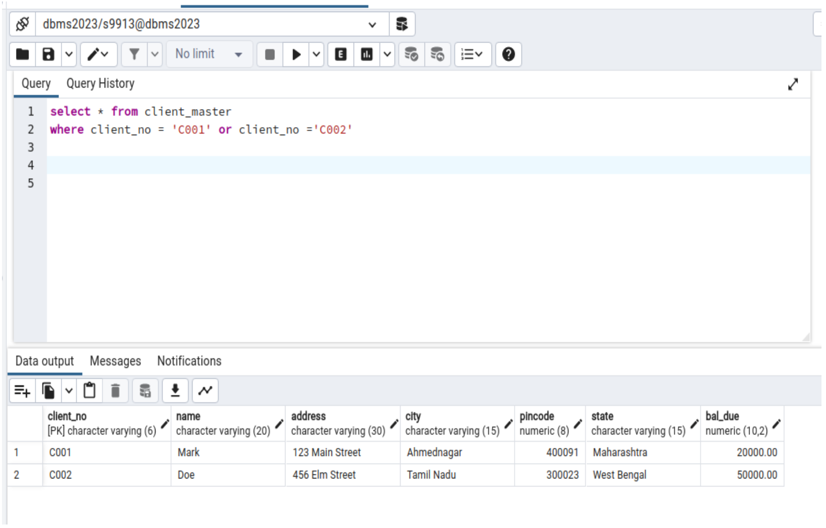
Query Query History

```sql
select * from sales_order
where substring(order_date::text from 6 for 2) = '01'
```

Data output Messages Notifications

	order_no character varying (6)	order_date date	client_no character varying (6)	dely_addr character varying (25)	salesman_no character varying (6)	dely_type character (1)	billed_yn character (1)	dely_date date	order_status character varyin

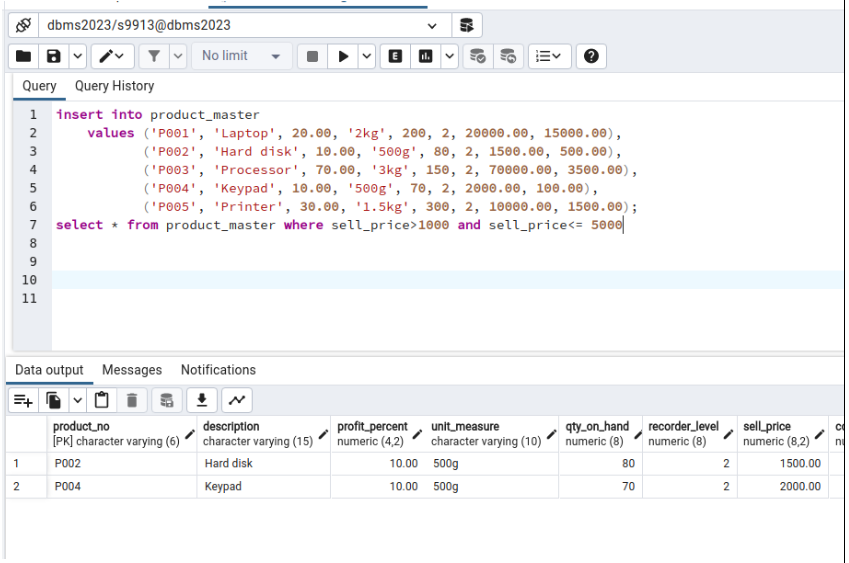
8. Display the order information for client_no C001 and C002



```sql
select * from client_master
where client_no = 'C001' or client_no ='C002'
```

	client_no [PK] character varying (6)	name character varying (20)	address character varying (30)	city character varying (15)	pincode numeric (8)	state character varying (15)	bal_due numeric (10,2)
1	C001	Mark	123 Main Street	Ahmednagar	400091	Maharashtra	20000.00
2	C002	Doe	456 Elm Street	Tamil Nadu	300023	West Bengal	50000.00

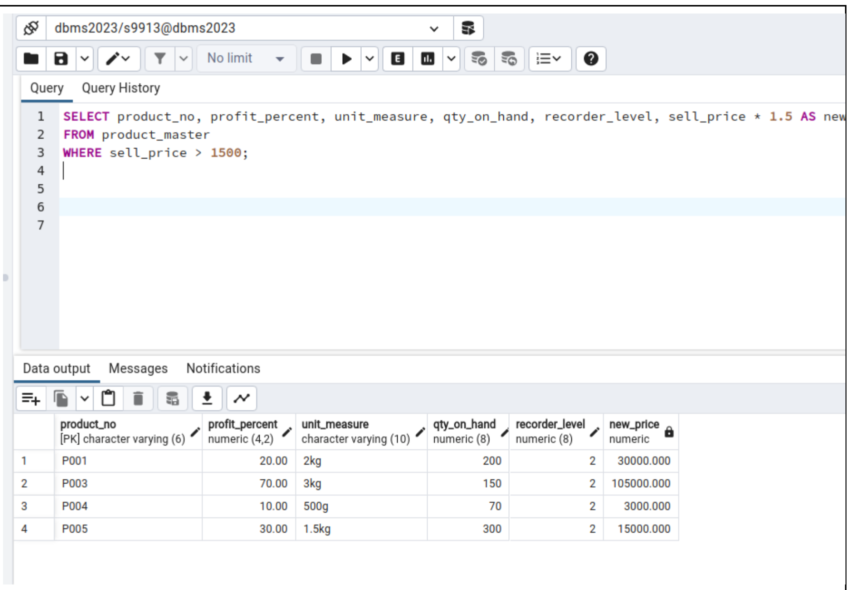
9. Find the products whose selling price is greater than 2000 and less than or equal to 5000



```sql
insert into product_master
    values ('P001', 'Laptop', 20.00, '2kg', 200, 2, 20000.00, 15000.00),
           ('P002', 'Hard disk', 10.00, '500g', 80, 2, 1500.00, 500.00),
           ('P003', 'Processor', 70.00, '3kg', 150, 2, 70000.00, 3500.00),
           ('P004', 'Keypad', 10.00, '500g', 70, 2, 2000.00, 100.00),
           ('P005', 'Printer', 30.00, '1.5kg', 300, 2, 10000.00, 1500.00);
select * from product_master where sell_price>1000 and sell_price<= 5000
```

	product_no [PK] character varying (6)	description character varying (15)	profit_percent numeric (4,2)	unit_measure character varying (10)	qty_on_hand numeric (8)	recorder_level numeric (8)	sell_price numeric (8,2)	co nu
1	P002	Hard disk	10.00	500g	80	2	1500.00	
2	P004	Keypad	10.00	500g	70	2	2000.00	

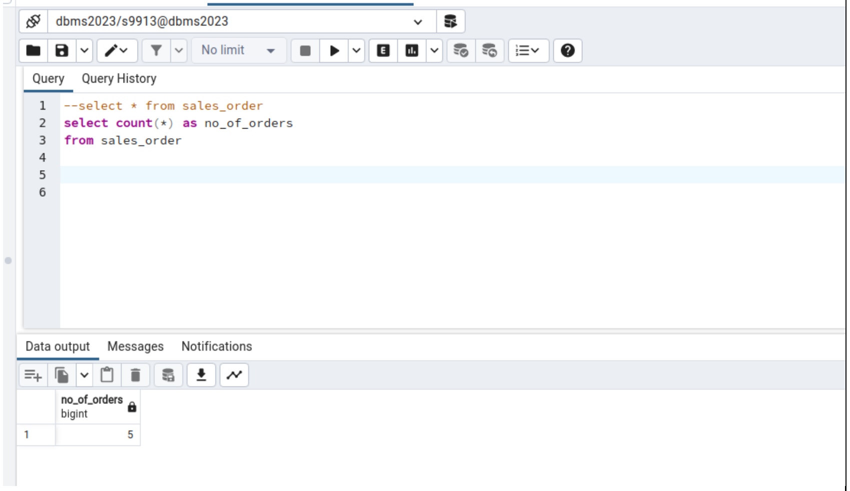
10. Find the products whose selling price is more than 1500. Calculate new selling price as original selling price * 1.5. Rename the new column in the above query as new_price


dbms2023/s9913@dbms2023

```
SELECT product_no, profit_percent, unit_measure, qty_on_hand, recorder_level, sell_price * 1.5 AS new
FROM product_master
WHERE sell_price > 1500;
```

	product_no [PK] character varying (6)	profit_percent numeric (4,2)	unit_measure character varying (10)	qty_on_hand numeric (8)	recorder_level numeric (8)	new_price numeric
1	P001	20.00	2kg	200	2	30000.000
2	P003	70.00	3kg	150	2	105000.000
3	P004	10.00	500g	70	2	3000.000
4	P005	30.00	1.5kg	300	2	15000.000

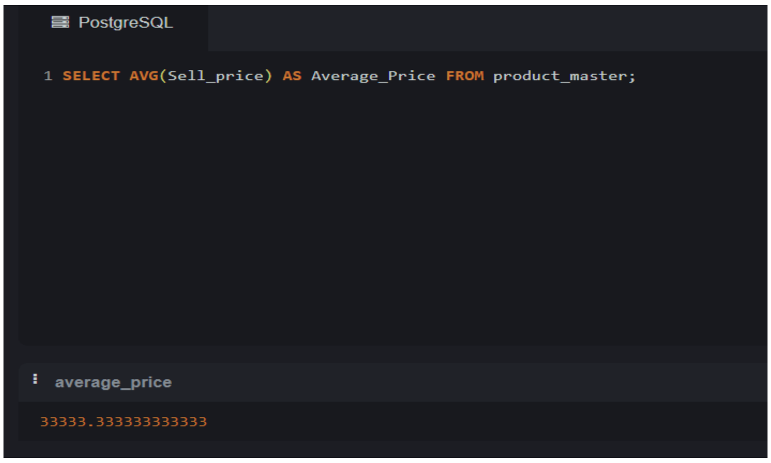
11. Count the total number of orders


dbms2023/s9913@dbms2023

```
--select * from sales_order
select count(*) as no_of_orders
from sales_order
```

	no_of_orders bigint
1	5

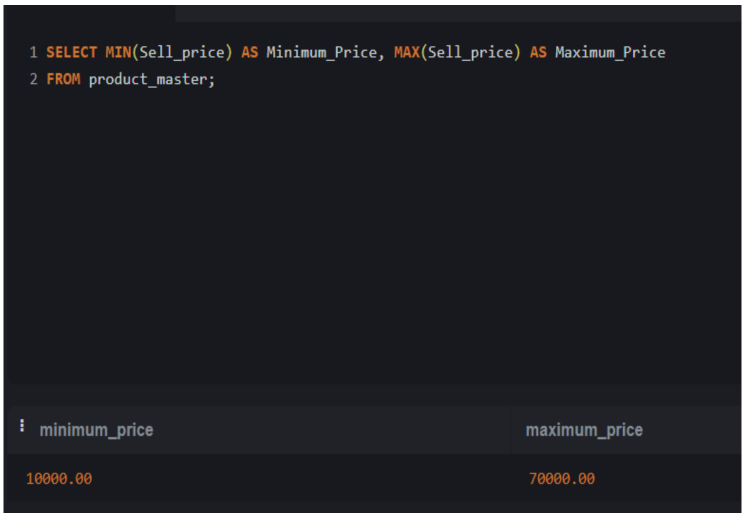
12. Calculate the average price of all the product


PostgreSQL

```
SELECT AVG(Sell_price) AS Average_Price FROM product_master;
```

```
average_price
33333.333333333333
```

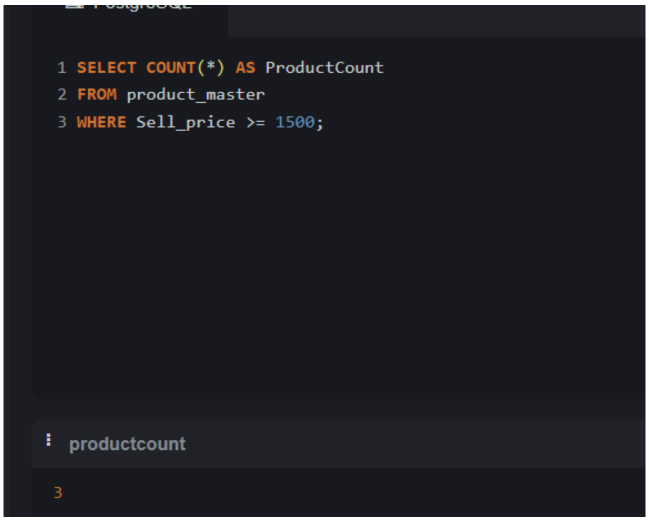
13. Determine minimum and maximum product prices



```
SELECT MIN(Sell_price) AS Minimum_Price, MAX(Sell_price) AS Maximum_Price
FROM product_master;
```

```
minimum_price          maximum_price
10000.00               70000.00
```

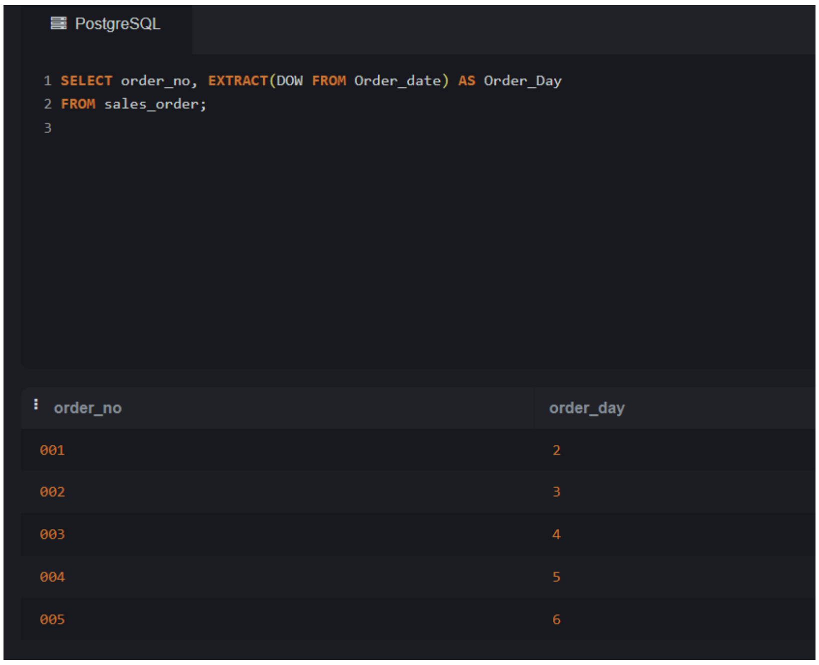
14. count the number of products having price greater than or equal to 1500



```
SELECT COUNT(*) AS ProductCount
FROM product_master
WHERE Sell_price >= 1500;
```

```
productcount
3
```

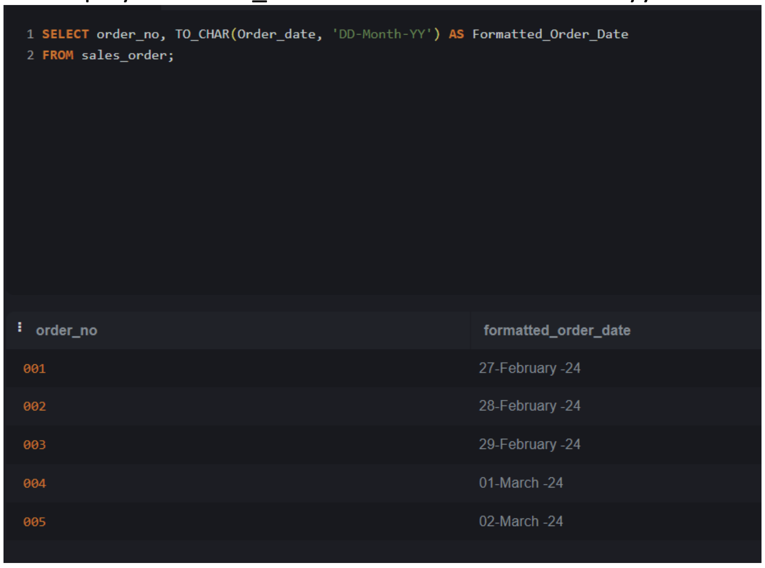
15. Display the order number and day on which clients placed their order



```
SELECT order_no, EXTRACT(DOW FROM Order_date) AS Order_Day
FROM sales_order;
```

```
order_no          order_day
001               2
002               3
003               4
004               5
005               6
```

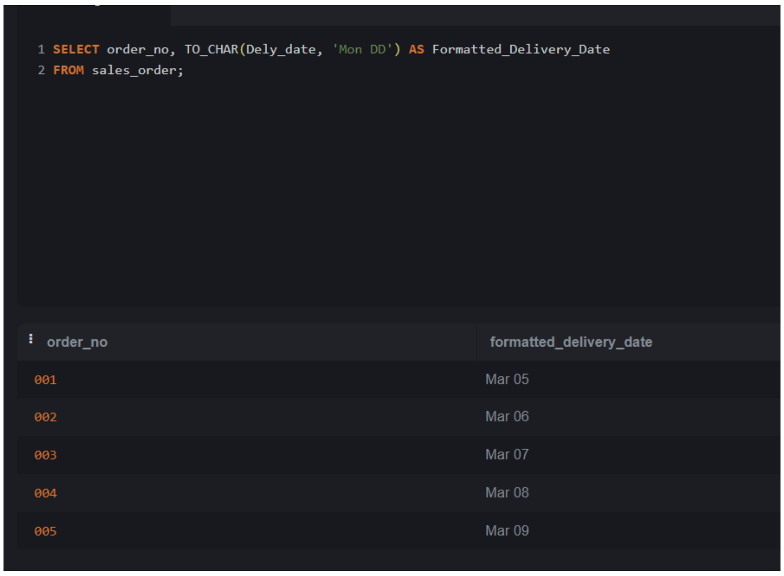
16. Display the order_date in the format ‘dd-month-yy’



```
SELECT order_no, TO_CHAR(Order_date, 'DD-Month-YY') AS Formatted_Order_Date
FROM sales_order;
```

```
order_no    formatted_order_date
001         27-February -24
002         28-February -24
003         29-February -24
004         01-March -24
005         02-March -24
```

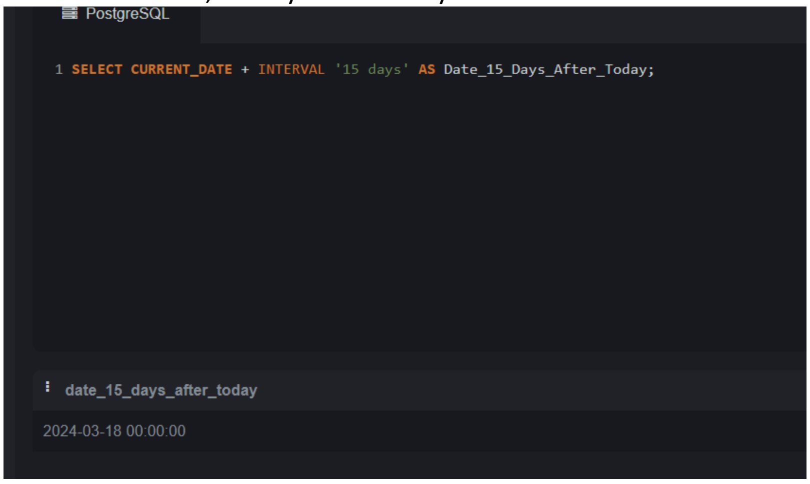
17. Display the month (in alphabets) and date when the order must be delivered



```
SELECT order_no, TO_CHAR(Dely_date, 'Mon DD') AS Formatted_Delivery_Date
FROM sales_order;
```

```
order_no    formatted_delivery_date
001         Mar 05
002         Mar 06
003         Mar 07
004         Mar 08
005         Mar 09
```

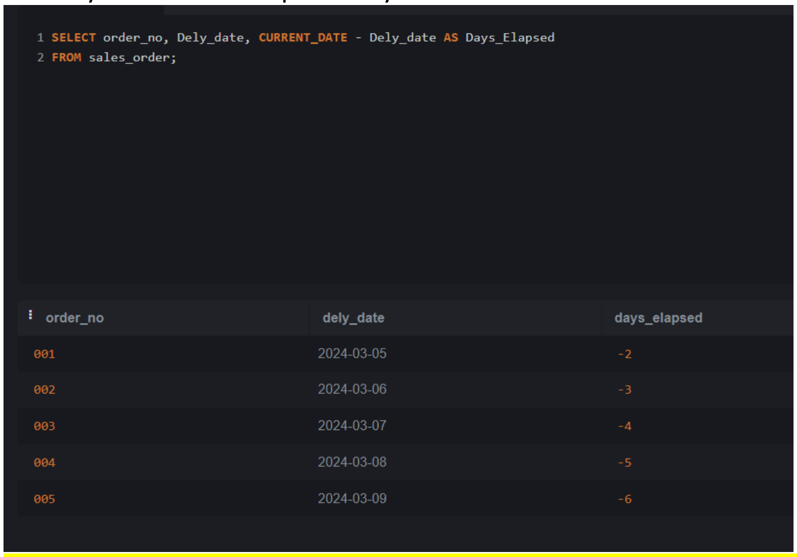
18. Find the date, 15 days after today’s date



```
SELECT CURRENT_DATE + INTERVAL '15 days' AS Date_15_Days_After_Today;
```

```
date_15_days_after_today
2024-03-18 00:00:00
```

19. Find the no. of days elapsed between today's date and the delivery date of orders placed by the clients.



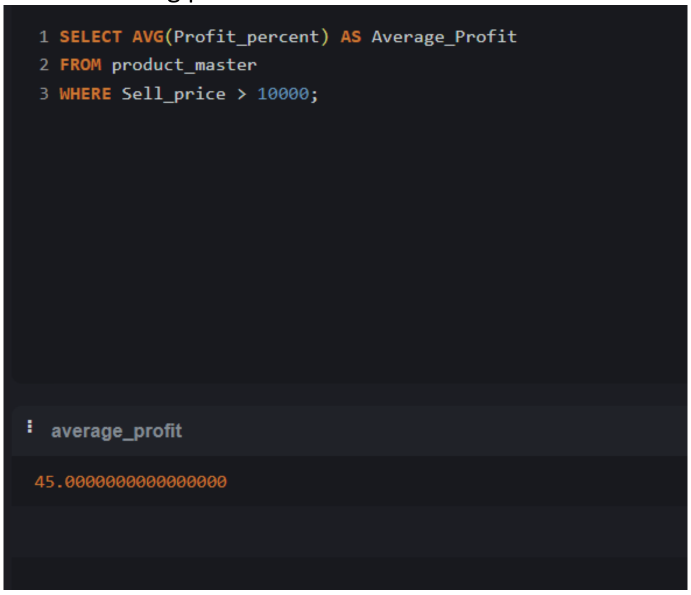
```
SELECT order_no, Dely_date, CURRENT_DATE - Dely_date AS Days_Elapsed
FROM sales_order;
```

```
order_no    dely_date     days_elapsed
001         2024-03-05    -2
002         2024-03-06    -3
003         2024-03-07    -4
004         2024-03-08    -5
005         2024-03-09    -6
```

Task2: Use select with where statement with SQL aggregate functions for the tables created in Expt. no. 3/mini project

1.product-master

To find the avg profit %

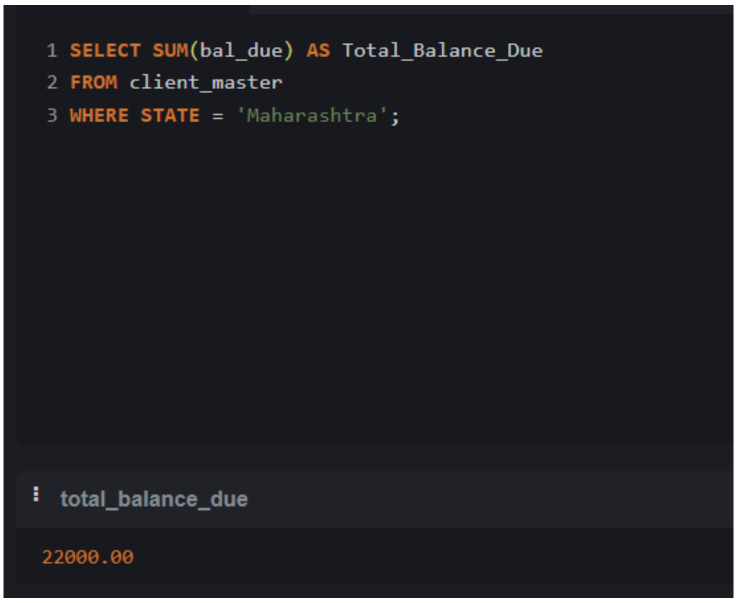


```
SELECT AVG(Profit_percent) AS Average_Profit
FROM product_master
WHERE Sell_price > 10000;
```

```
average_profit
45.0000000000000000
```

2.cliient master

to find the total balance due for clients in the state of Maharashtra.

```sql
SELECT SUM(bal_due) AS Total_Balance_Due
FROM client_master
WHERE STATE = 'Maharashtra';
```

```
total_balance_due
22000.00
```

Post Lab Questions:

1. Write a short note on DBA
   A Database Administrator (DBA) is a professional responsible for designing, implementing, and managing database systems. They handle tasks like database installation, performance optimization, security management, backup and recovery, and ensure overall efficiency and reliability of the database.

2. Explain system structure of DBMS
   The system structure of a Database Management System (DBMS) includes users, applications, the DBMS itself, and key components such as the database engine, query processor, transaction manager, storage manager, buffer manager, data dictionary, and database files. These components work together to manage data storage, retrieval, and ensure data integrity and security.

3. Write different date functions

```
SELECT CURRENT_DATE;
SELECT CURRENT_TIME;
SELECT CURRENT_TIMESTAMP;
SELECT DATE_FORMAT(NOW(), '%Y-%m-%d') AS FormattedDate;
SELECT EXTRACT(MONTH FROM hire_date) AS HireMonth FROM employees;
```

4. Differentiate between group by and having with example
   GROUP BY: Used to group rows based on specified columns and apply aggregate functions to each group.

```
SELECT department_id, COUNT(*) AS EmployeeCount
FROM employees
GROUP BY department_id;
```

   HAVING: Used to filter the results of a `GROUP BY` query based on conditions involving aggregate functions.

```
SELECT department_id, COUNT(*) AS EmployeeCount
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5;
```

5. Give different string functions
   CONCAT:

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM employees;
```

   UPPER and LOWER:

```
SELECT UPPER(last_name) AS UpperCaseLastName
FROM employees;
```

   LENGTH:

```
SELECT LENGTH(email) AS EmailLength
```

		FROM employees;
		SUBSTRING:
		SELECT SUBSTRING(last_name, 1, 3) AS Initials
		FROM employees;
		CONVERT:
		SELECT CAST('123' AS INT) AS ConvertedNumber;