| SE-COMP Roll number :9914 | | | |
| --- | --- | --- | --- |
| Experiment no. : 5 Date of Implementation : 03/03/2024 | | | |
| 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 |   2. Insert 5-6 records in table.  3. Find the names of all clients having ‘a’ as the second letter in their names.  4. Find out the clients who stay in a city whose second letter is ‘a’  5. Find the list of all clients who stay in ‘mumbai’ ordered by their names  6. Print the list of clients whose bal\_due is greater than value 10000  7. Print the information from sales\_order table for orders placed in the month of January  8. Display the order information for client\_no C001 and C002  9. Find the products whose selling price is greater than 2000 and less than or equal to 5000  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  11. Count the total number of orders  12. Calculate the average price of all the product  13. Determine minimum and maximum product prices  14. count the number of products having price greater than or equal to 1500  15. Display the order number and day on which clients placed their order  16. Display the order\_date in the format ‘dd-month-yy’  17. Display the month (in alphabets) and date when the order must be delivered  18. Find the date, 15 days after today’s date  19. Find the no. of days elapsed between today’s date and the delivery date of orders placed by the clients.  Task2: Use select with where statement with SQL aggregate functions for the tables created in Expt. no. 3/mini project |
|  |  |
|  | **Post Lab Questions:** | 1. Write a short note on DBA 2. Explain system structure of DBMS 3. Write different date functions 4. Differentiate between group by and having with example 5. Give different string functions |

Task 1:

Q1.

create table sales\_order(

order\_no varchar(6),

Order\_date date,

Client\_no varchar(6),

Dely\_addr varchar(25),

Salesman\_no varchar(6),

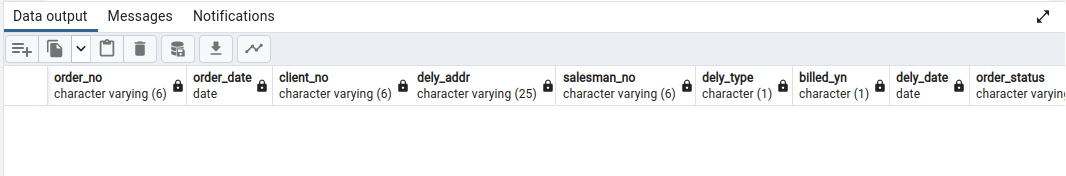
Dely\_type char(1),

Billed\_yn char(1),

Dely\_date date,

Order\_status varchar(10));

select \* from sales\_order;



Q2.

insert into sales\_order

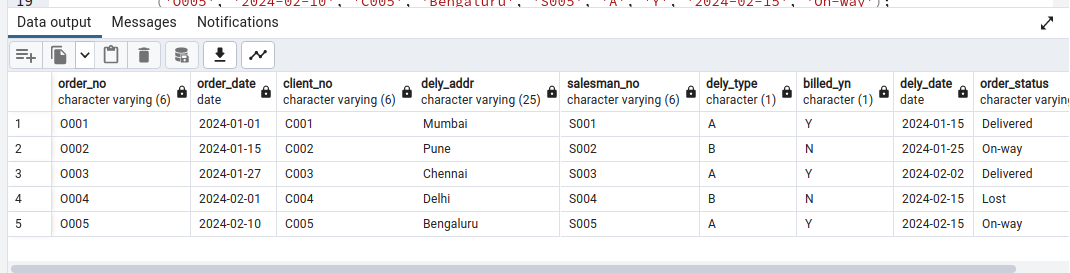
values ('O001', '2024-01-01', 'C001', 'Mumbai', 'S001', 'A', 'Y', '2024-01-15', 'Delivered'),

('O002', '2024-01-15', 'C002', 'Pune', 'S002', 'B', 'N', '2024-01-25', 'On-way'),

('O003', '2024-01-27', 'C003', 'Chennai', 'S003', 'A', 'Y', '2024-02-02', 'Delivered'),

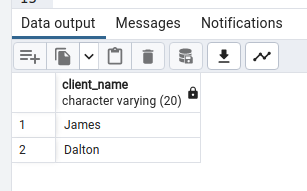
('O004', '2024-02-01', 'C004', 'Delhi', 'S004', 'B', 'N', '2024-02-15', 'Lost'),

('O005', '2024-02-10', 'C005', 'Bengaluru', 'S005', 'A', 'Y', '2024-02-15', 'On-way');



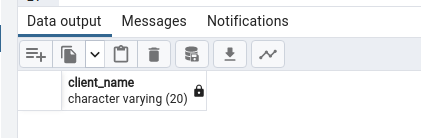
Q3.

select client\_name from client\_master WHERE SUBSTR(client\_name, 2, 1) = 'a';



Q4.

select client\_name from client\_master WHERE SUBSTR(city, 2, 1) = 'a';

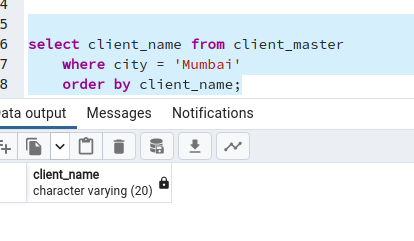


Q5.

select client\_name from client\_master

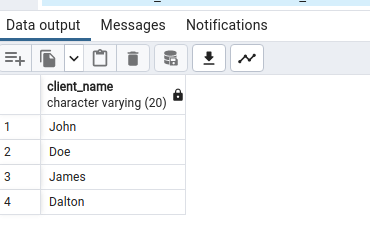
where city = 'Mumbai'

order by client\_name;



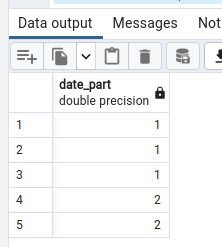
Q6.

select client\_name from client\_master where bal\_due>10000;



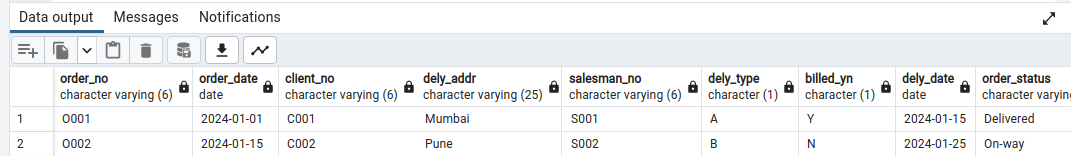
Q7.

selectextract(month from Order\_date) from sales\_order;



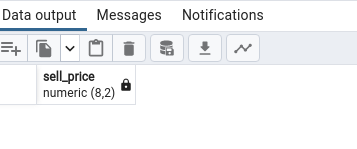
Q8.

select \* from sales\_order where Client\_no = 'C001' or Client\_no = 'C002';



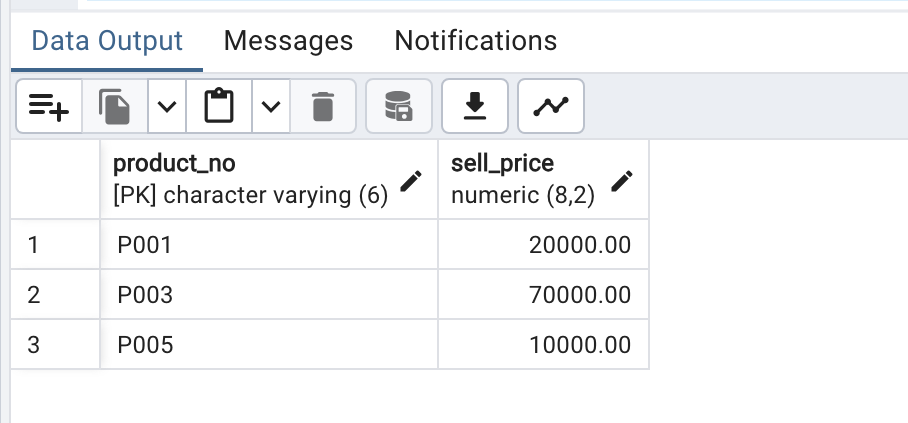
Q9.

select Sell\_price from Product\_master where Sell\_price > 2000.00 and Sell\_price <=5000.00;



Q10.

select product\_no, Sell\_price from Product\_master where Sell\_price > 1500.00;

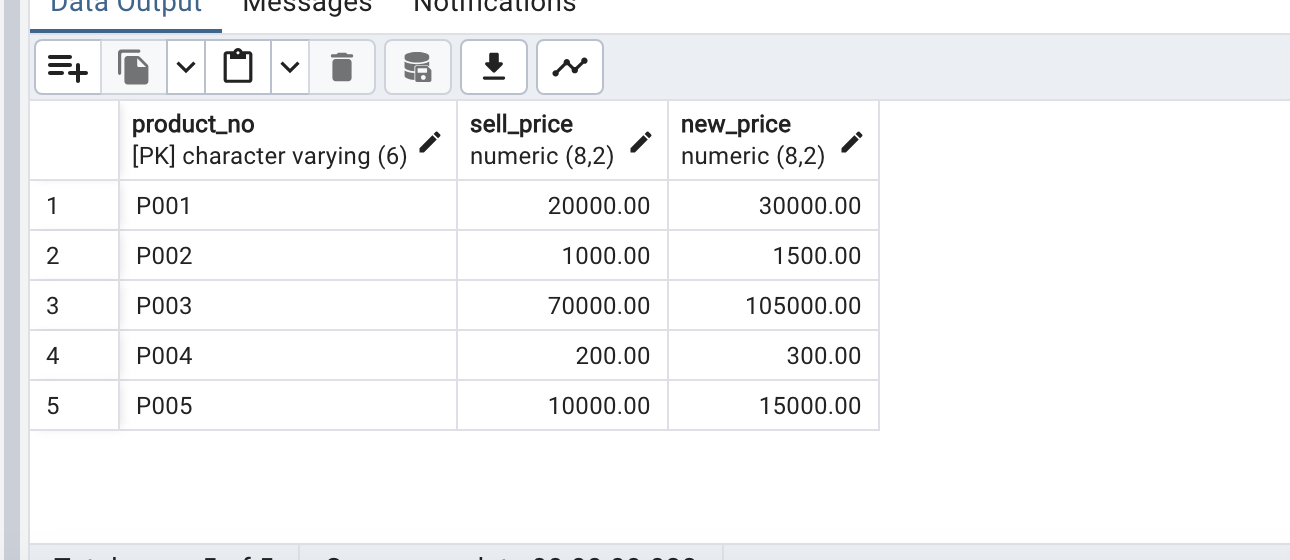


alter table Product\_master

add column new\_price numeric(8, 2);

update Product\_master

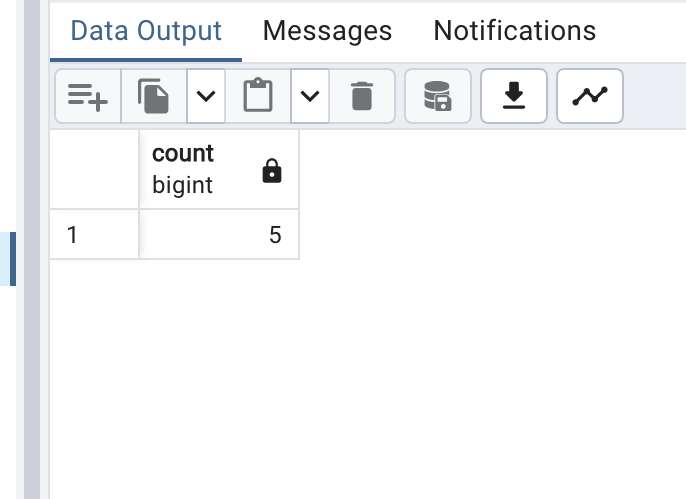
set new\_price = Sell\_price \* 1.5;



Q11.

select count(order\_no)

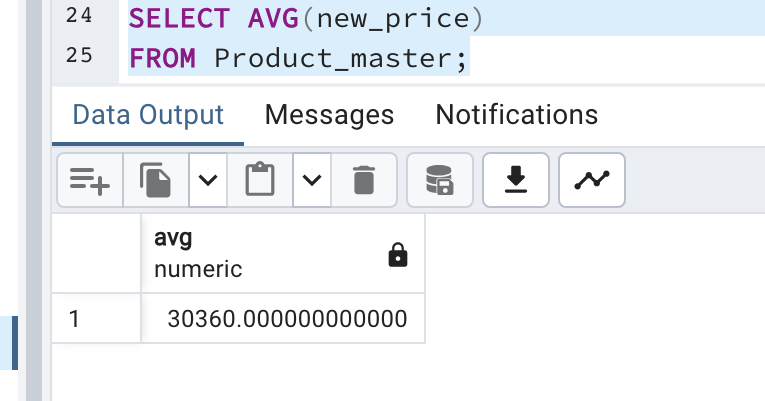
from sales\_order;



Q12.

SELECT AVG(new\_price)

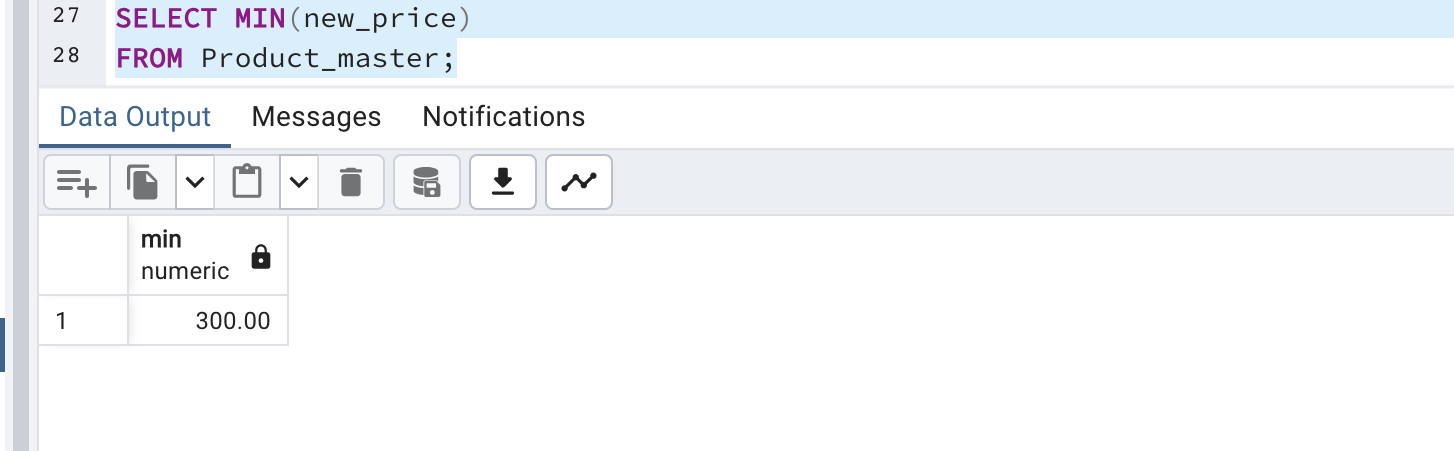
FROM Product\_master;



Q13.

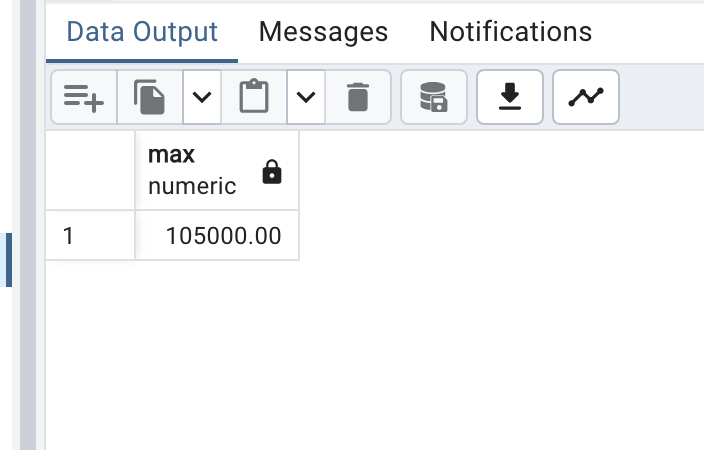
SELECT MIN(new\_price)

FROM Product\_master;



SELECT max(new\_price)

FROM Product\_master;

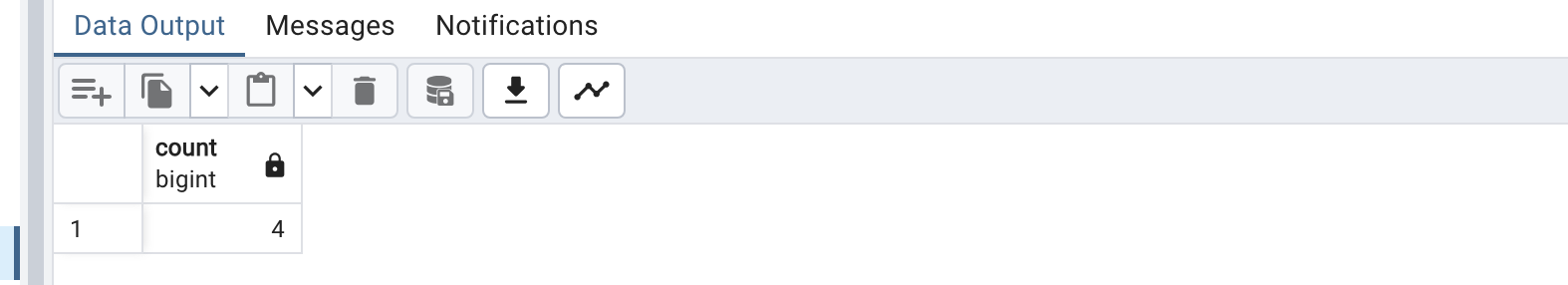


Q14.

select count(product\_no)

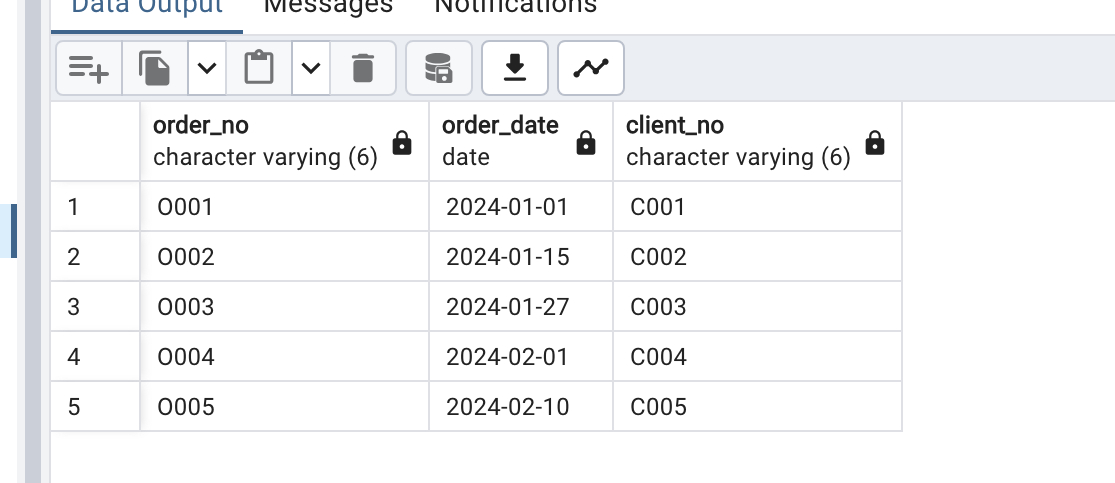
from Product\_master

where new\_price >= 1500.00;



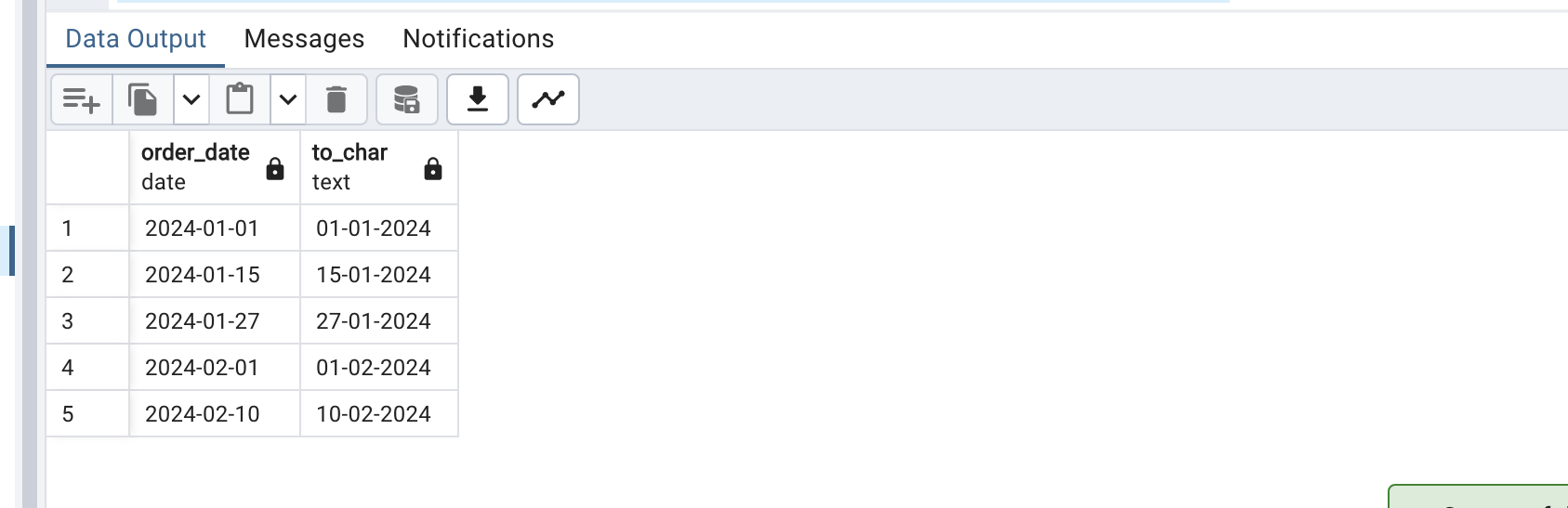
Q15.

select order\_no, Order\_date, Client\_no from sales\_order;



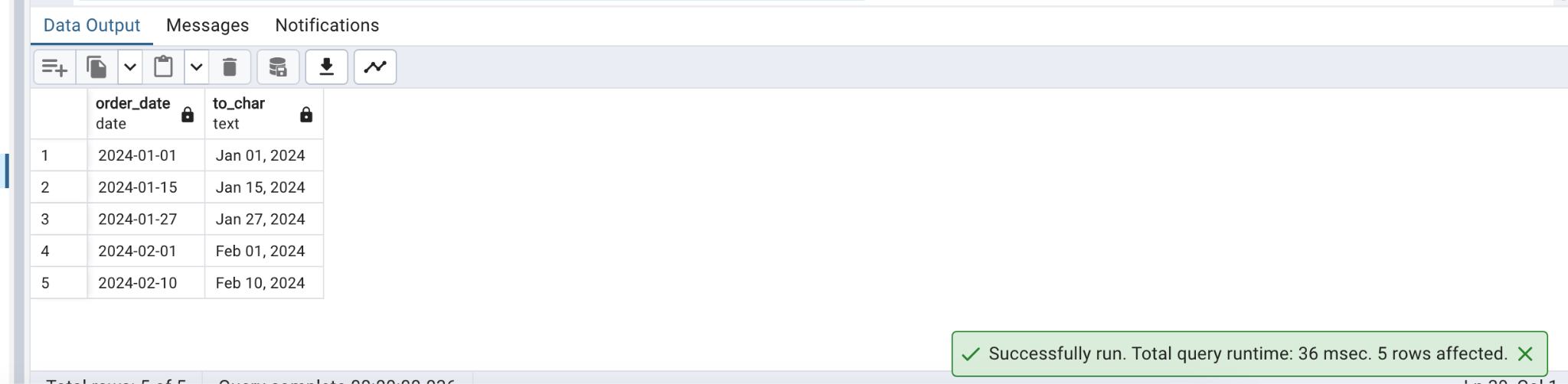
Q16.

select Order\_date, to\_char(Order\_date, 'DD-MM-YYYY') from sales\_order;



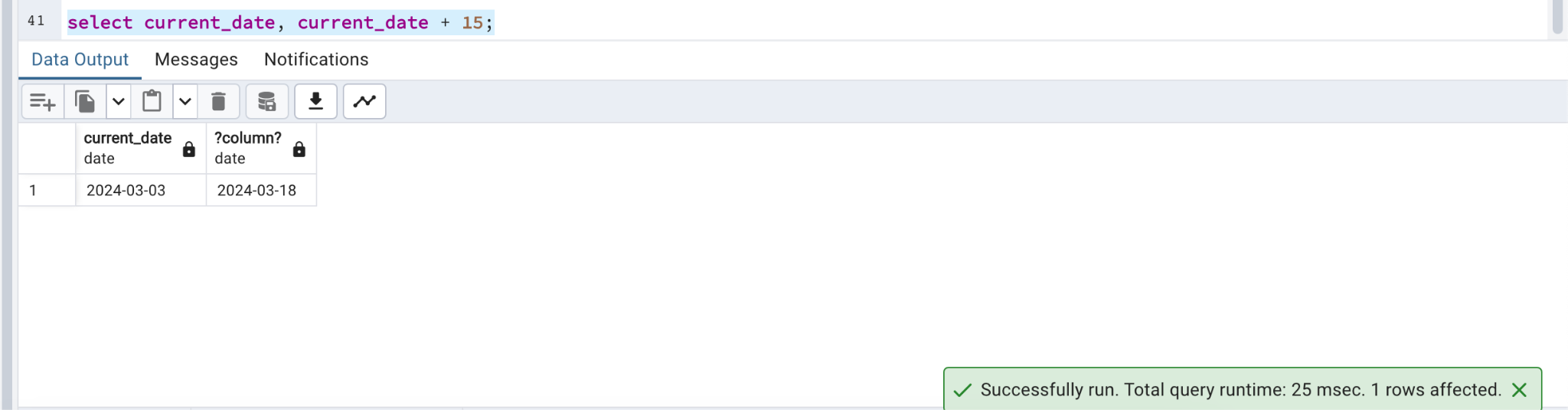
Q17.

select Order\_date, to\_char(Order\_date, 'Mon dd, yyyy') from sales\_order;



Q18.

select current\_date, current\_date + 15;



Q19.

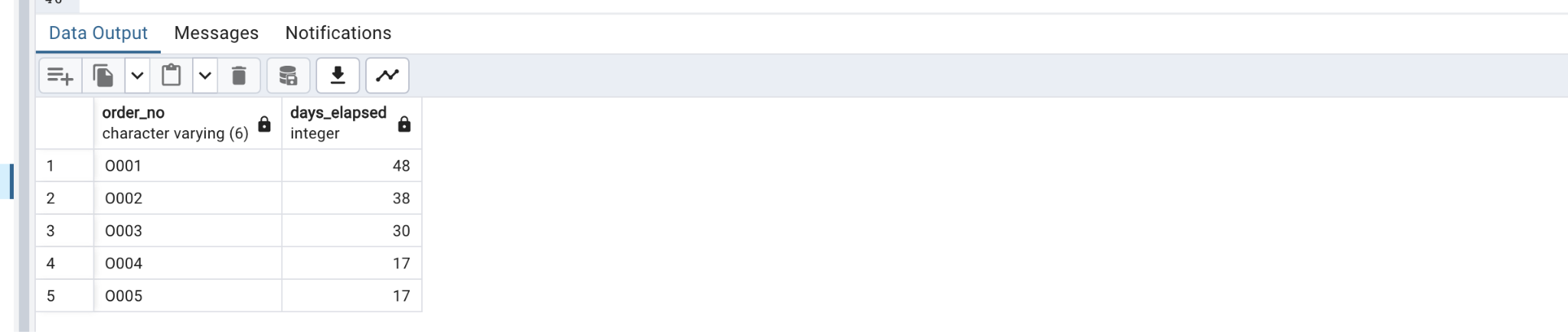
SELECT

order\_no,

current\_date - Dely\_date AS days\_elapsed

FROM

sales\_order;



Task 2:

1. Retrieve the maximum total amount for invoices from a specific branch:

SELECT

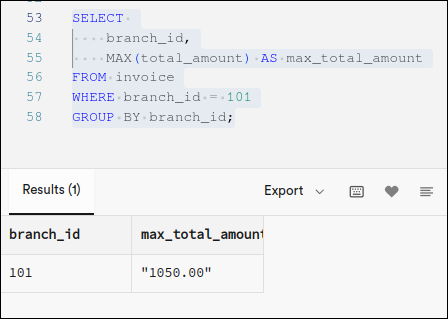
branch\_id,

MAX(total\_amount) AS max\_total\_amount

FROM invoice

WHERE branch\_id = 101

GROUP BY branch\_id;



1. Retrieve the gross amount and total taxes for invoice with a specific mode of payment

SELECT

mode\_of\_payment,

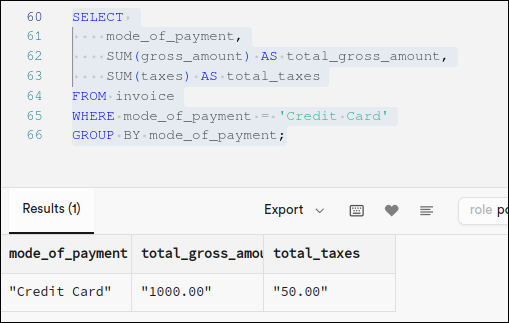
SUM(gross\_amount) AS total\_gross\_amount,

SUM(taxes) AS total\_taxes

FROM invoice

WHERE mode\_of\_payment = 'Credit Card'

GROUP BY mode\_of\_payment;

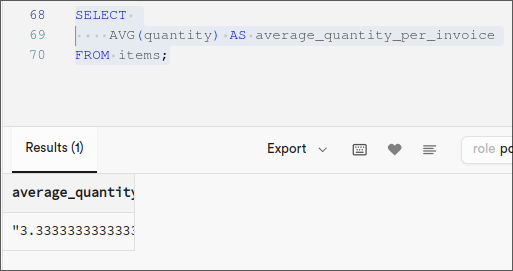


1. Find the average quantity of item per invoice

SELECT

AVG(quantity) AS average\_quantity\_per\_invoice

FROM items;



1. Count the number of invoices for each invoice type:

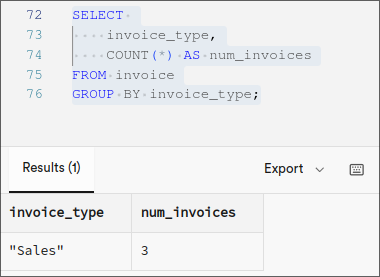
SELECT

invoice\_type,

COUNT(\*) AS num\_invoices

FROM invoice

GROUP BY invoice\_type;



1. Find the branch with the highest total amount of invoices:

SELECT

branch\_id,

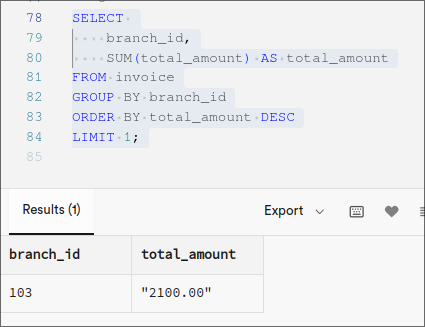
SUM(total\_amount) AS total\_amount

FROM invoice

GROUP BY branch\_id

ORDER BY total\_amount DESC

LIMIT 1;



1. Retrieve the total quantity of items sold for a specific invoice type:

SELECT

invoice\_type,

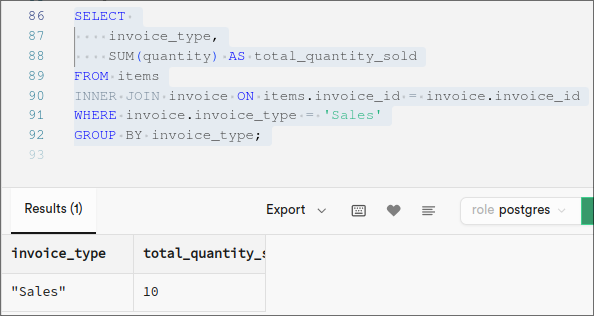
SUM(quantity) AS total\_quantity\_sold

FROM items

INNER JOIN invoice ON items.invoice\_id = invoice.invoice\_id

WHERE invoice.invoice\_type = 'Sales'

GROUP BY invoice\_type;



Postlab:

Q1.

In the field of data management system, Database Administrators (DBAs) play a critical role. Their responsibilities involves the following tasks:

* Installation and configuration of database systems
* Implementing robust security measures to protect sensitive information
* Regular backups and recovery procedures to safeguard against data loss
* Performance optimization to maintain efficient data access and retrieval
* User management and access control to ensure appropriate data permissions

Q2.

The system structure of DBMS is as follows:

* Users: Interact with the system (individuals, applications).
* DBMS Engine: Manages data storage, retrieval, and manipulation (core component). Languages:
* DDL: Defines database structure (creating tables).
* DML: Performs data operations (adding, updating, deleting, querying).
* Data Dictionary: Stores metadata about the database (structure, relationships, permissions).
* Transaction Management: Ensures data integrity (ACID properties).
* Storage Management: Efficiently stores and retrieves data.
* Query Optimization: Improves data retrieval performance.
* Concurrency Control: Manages concurrent access without conflicts.

Q3.

For PostgreSQL, commonly used date functions include:

- DATE\_ADD(): Adds a specified time interval to a date.

- DATEDIFF(): Returns the difference between two dates.

- DATEPART(): Extracts a specific part (such as year, month, day) from a date.

- CURRENT\_DATE: Returns the current date.

- TO\_CHAR(): Formats a date value as per a specified format.

- EXTRACT(): Extracts the day, month, year, etc., from a date.

- NOW(): Returns the current date and time.

Q4. ⇒

* GROUP BY groups rows based on common values in a specified column or columns, facilitating the aggregation of data. For example:

SELECT department, AVG(salary) AS avg\_salary FROM employees GROUP BY department\_name;

In this example, the rows in the 'employees' table are grouped by the 'department\_name' column, and the average salary for each department is calculated.

* HAVING filters groups based on aggregate conditions after the GROUP BY operation has been applied. For example:

SELECT department, AVG(salary) AS avg\_salary FROM employees GROUP BY department HAVING AVG(salary) > 10000;

In this example, the rows are first grouped by department, and then the HAVING clause filters out departments where the average salary is greater than Rs.10,000. Only groups that meet this condition will be included in the final result.

Q5. ⇒

String functions in SQL are used to perform operations on character strings. Some common string functions include:

CONCAT(): Concatenates two or more strings.

LENGTH() or LEN(): Returns the length of a string.

UPPER() or LOWER(): Converts a string to uppercase or lowercase.

SUBSTRING() or SUBSTR(): Extracts a substring from a string.

REPLACE(): Replaces occurrences of a substring within a string with another substring.

TRIM(): Removes leading and trailing spaces from a string.

LEFT() or RIGHT(): Returns a specified number of characters from the left or right of a string.

INSTR() or CHARINDEX(): Returns the position of a substring within a string.

REVERSE(): Reverses the characters in a string.

CONVERT() or CAST(): Converts a string from one data type to another.