SE-COMP(B)	Roll number : 8942
Experiment no. : 4	Date of Implementation : 26/02/2021
Aim : To implement simple SQL commands	
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 4	Basic SQL Commands
Aim	To implement simple SQL commands
Tools	PostgreSQL/MySql
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_name, column_name FROM table_name ORDER BY column_name ASC|DESC, column_name ASC|DESC;

Procedure

TASK 1:

1. Create following table:

Table name : **sales_order**

Column Name	Data type	Size	Constraint
order_no	varchar	6	Primary Key
Order_date	date		NOT NULL
Client_no	varchar	6	NOT NULL
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

Post Lab Questions:

1. Write a short note on DBA.
2. Write different date functions and date formats.
3. Differentiate between group by and having with example.
4. Give different string functions.

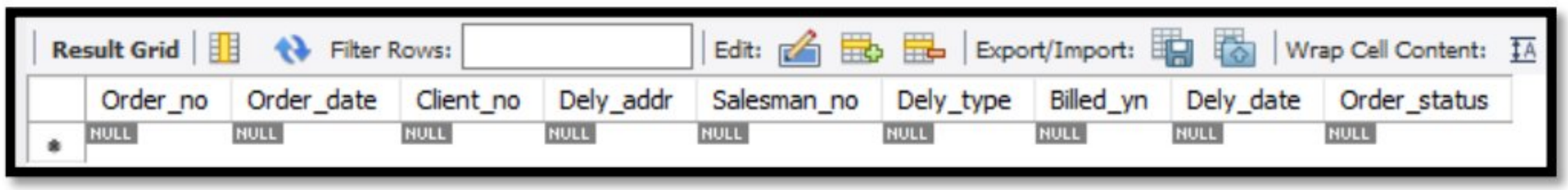
Task1

1.] Create table sales_order table

Query:

```
create table sales_order(
order_no varchar(6),
Order_date date NOT NULL,
Client_no varchar(6) NOT NULL,
Dely_addr varchar(25),
Salesman_no varchar(6),
Dely_type char(1),
Billed_yn char(1),
Dely_date date,
Order_status varchar(10), PRIMARY KEY(order_no));
```

Screenshot:



2. Insert 5-6 records in table.

Query:

```
Insert into sales_order value('101','2021-01-26', 'C001', 'Bandra', '1A', 's', 'y', '2021-01-17',
'delivered');
Insert into sales_order value('102','2020-12-21', 'C002', 'Ashoknagar', '2B', 's', 'y', '2020-12-15',
'delivered');
Insert into sales_order value('103','2021-02-02', 'C003', 'Bandra', '3C', 'r', 'n', '2021-01-09',
'delivered');
Insert into sales_order value('104','2021-01-06', 'C004', 'Ramnagar', '4D', 's', 'y', '2021-01-01',
'shipped');
Insert into sales_order value('105','2021-02-15', 'C005', 'Ghandhinagar', '5E', 'r', 'n', '2021-01-19',
'shipped');
Insert into sales_order value('106','2021-01-10', 'C006', 'Mira Road', '6F', 's', 'y', '2021-01-07',
'delivered');
Insert into sales_order value('107','2021-02-10', 'C007', 'Mira Road', '6F', 's', 'y', '2021-01-18',
'shipped');
```

Screenshot:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Order_no	Order_date	Client_no	Dely_addr	Salesman_no	Dely_type	Billed_yn	Dely_date	Order_status
101	2021-01-26	C001	Bandra	1A	s	y	2021-01-17	delivered
102	2020-12-21	C002	Ashoknagar	2B	s	y	2020-12-15	delivered
103	2021-02-02	C003	Bandra	3C	r	n	2021-01-09	delivered
104	2021-01-06	C004	Ramnagar	4D	s	y	2021-01-01	shipped
105	2021-02-15	C005	Ghandhinagar	5E	r	n	2021-01-19	shipped
106	2021-01-10	C006	Mira Road	6F	s	y	2021-01-07	delivered
107	2021-02-10	C007	Mira Road	6F	s	y	2021-01-18	shipped

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

Query:

select Name FROM client_master WHERE Name like '_a%';

Screenshot:

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

Name
Aarya

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

Query:

select Name FROM client_master WHERE City like '_a%';

Screenshot:

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

Name
Aarya
Ashley

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

Query:

select Name FROM client_master WHERE City like 'Mumbai';

Screenshot:

Name
Elwin
Elaine
Jerry

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

Query:

select Name FROM client_master WHERE Bal_due > 10000;

Screenshot:

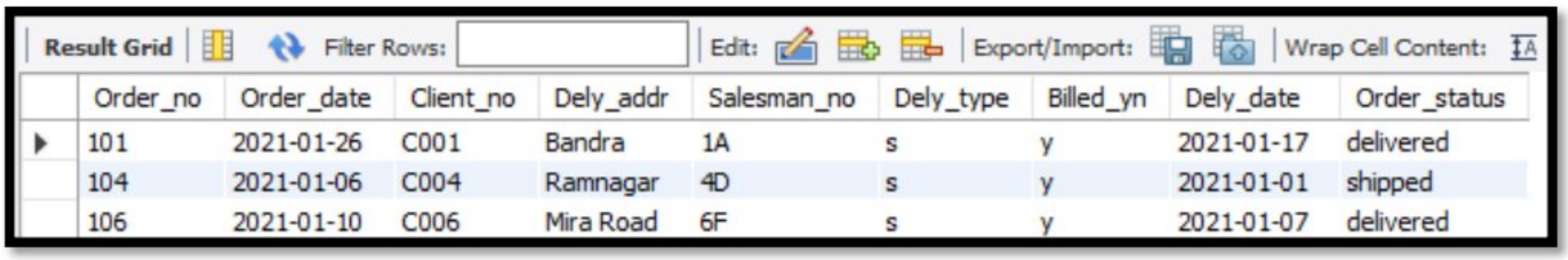
Name
Jerry
Aarya
Ashley

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

Query:

select * from sales_order where month(Order_date) = 01 ;

Screenshot:



Order_no	Order_date	Client_no	Dely_addr	Salesman_no	Dely_type	Billed_yn	Dely_date	Order_status
101	2021-01-26	C001	Bandra	1A	s	y	2021-01-17	delivered
104	2021-01-06	C004	Ramnagar	4D	s	y	2021-01-01	shipped
106	2021-01-10	C006	Mira Road	6F	s	y	2021-01-07	delivered

8. Display the order information for client_no C001 and C002

Query:

select * from sales_order where (Client_no = 'C001') or (Client_no = 'C002');

Screenshot:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Order_no	Order_date	Client_no	Dely_addr	Salesman_no	Dely_type	Billed_yn	Dely_date	Order_status
101	2021-01-26	C001	Bandra	1A	s	y	2021-01-17	delivered
102	2020-12-21	C002	Ashoknagar	2B	s	y	2020-12-15	delivered

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

Query:

select description FROM product_master WHERE sell_price > 2000 and sell_price <= 5000;

Screenshot:

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

description
Keyboard
Hard disk

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

Query:

select description ,
sell_price*1.5 "new_price" from product_master where sell_price > 1500;

Screenshot:

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

description	new_price
Monitor	60000.000
Keyboard	3750.000
Hard disk	7500.000

11. Count the total number of orders

Query:

select COUNT(*) "TOTAL ORDERS" from sales_order;

Screenshot:

TOTAL ORDERS
7

12. Calculate the average price of all the product

Query:

select AVG(sell_price), AVG(cost_price) from product_master;

Screenshot:

AVG(sell_price)	AVG(cost_price)
9822.000000	3050.000000

13. Determine minimum and maximum product prices

Query:

select MIN(sell_price), MAX(sell_price) from product_master;

Screenshot:

MIN(sell_price)	MAX(sell_price)
110.00	40000.00

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

Query:

select COUNT(*) from product_master where sell_price >= 1500;

Screenshot:

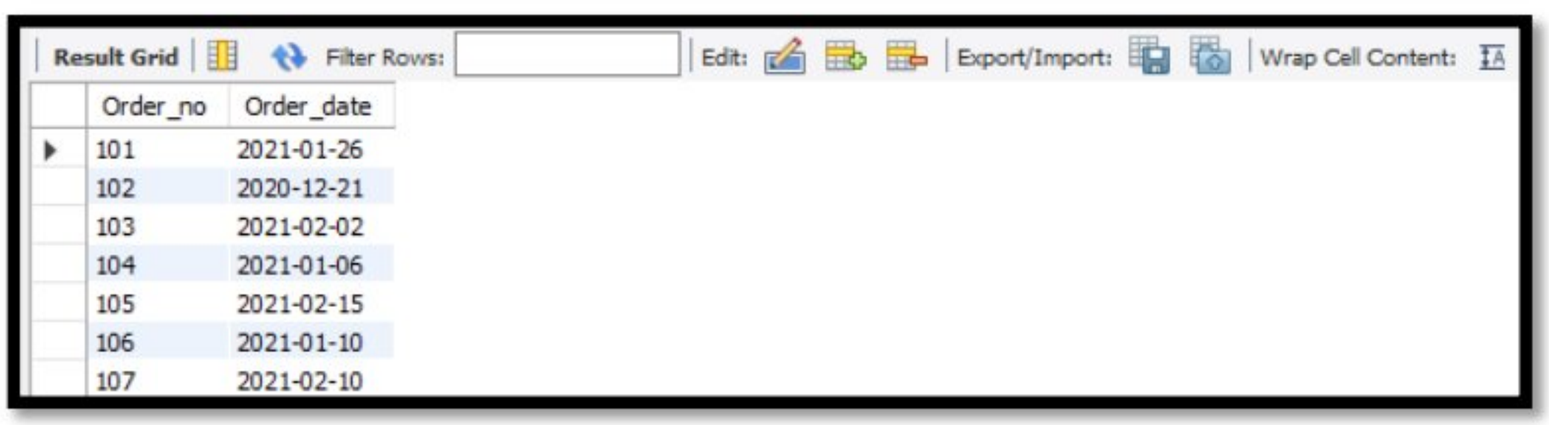
COUNT(*)
4

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

Query:

select Order_no, Order_date from sales_order;

Screenshot:



Order_no	Order_date
101	2021-01-26
102	2020-12-21
103	2021-02-02
104	2021-01-06
105	2021-02-15
106	2021-01-10
107	2021-02-10

16. Display the order_date in the format 'dd-month-yy'

Query:

select date_format(Order_date, '%d-%M-%Y') from sales_order;

Screenshot:

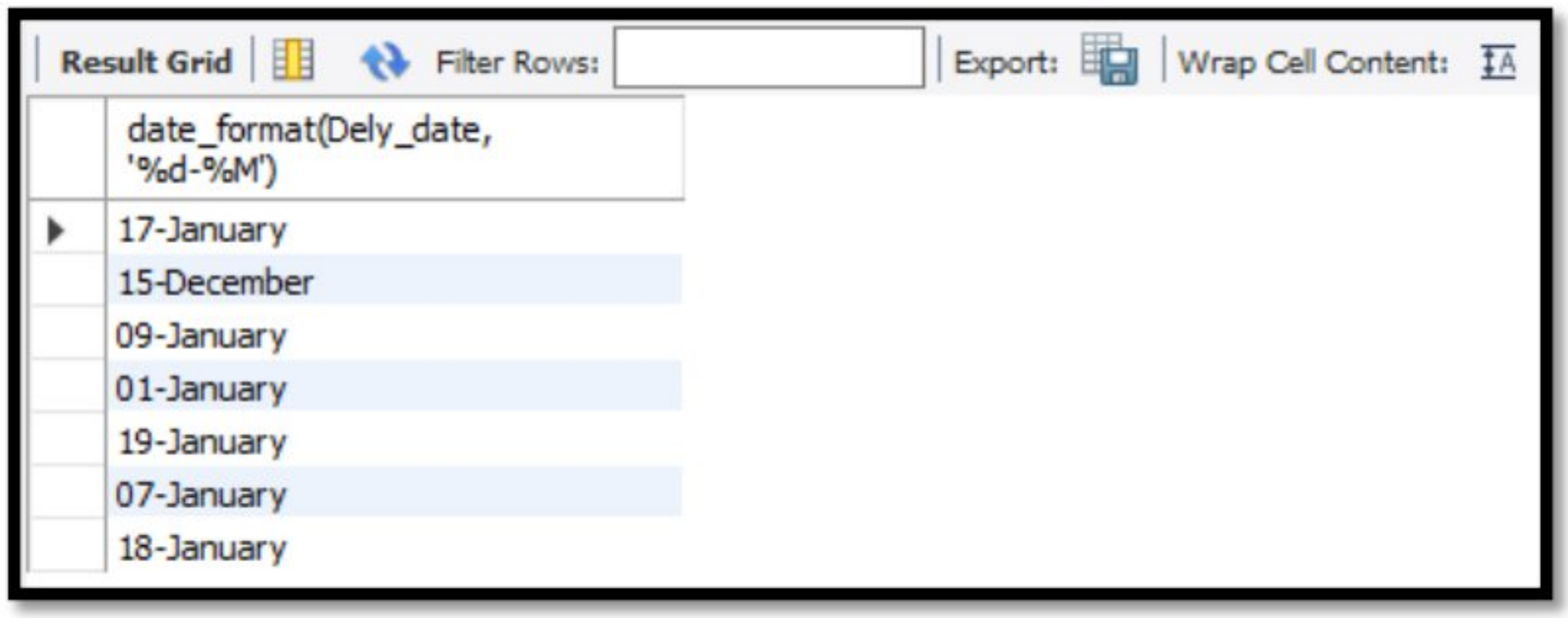
date_format(Order_date, '%d-%M-%Y')
26-January-2021
21-December-2020
02-February-2021
06-January-2021
15-February-2021
10-January-2021
10-February-2021

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

Query:

select date_format(Dely_date, '%d-%M') from sales_order;

Screenshot:



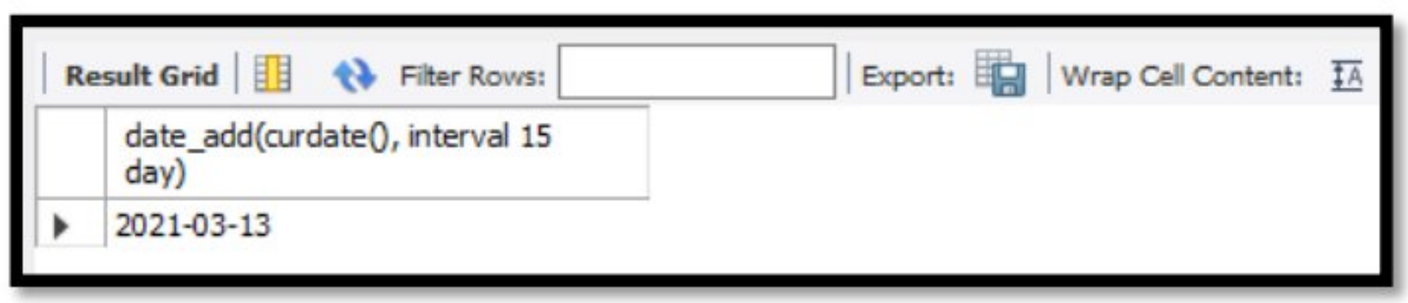
date_format(Dely_date, '%d-%M')
17-January
15-December
09-January
01-January
19-January
07-January
18-January

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

Query:

select date_add(curdate(), interval 15 day);

Screenshot:



date_add(curdate(), interval 15 day)
2021-03-13

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

Query:

select datediff(curdate(),Dely_date) from sales_order;

Screenshot:

datediff(curdate(),Dely_date)
40
73
48
56
38
50
39

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

- **AVG() - Returns the average value**

Query:

select AVG(sell_price), AVG(cost_price) from product_master;

Screenshot:

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

AVG(sell_price)	AVG(cost_price)
9822.000000	3050.000000

- **COUNT() - Returns the number of rows**

Query:

select COUNT(*) from product_master where sell_price >= 1500;

Screenshot:

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

COUNT(*)
4

- **FIRST() - Returns the first value**

Query:

select * from product_master order by product_no ASC LIMIT 1;

Screenshot:

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

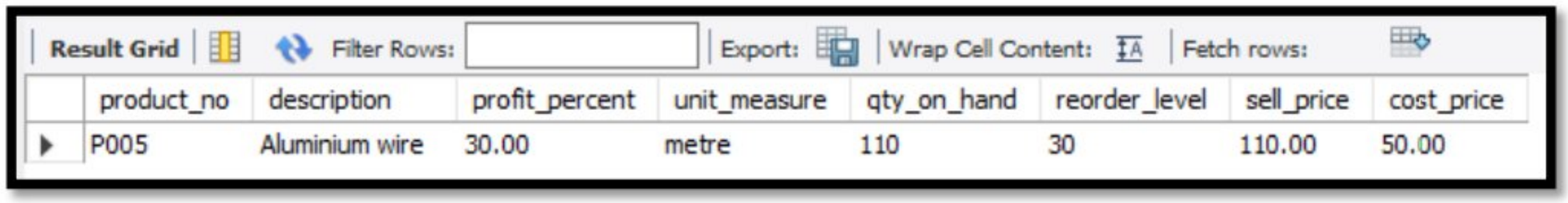
product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	sell_price	cost_price
P001	Monitor	10.00	inches	300	30	40000.00	10000.00

- **LAST() - Returns the last value**

Query:

select * from product_master order by product_no DESC LIMIT 1;

Screenshot:



product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	sell_price	cost_price
P005	Aluminium wire	30.00	metre	110	30	110.00	50.00

- **MAX() - Returns the largest value**

Query:

select MAX(sell_price) from product_master;

Screenshot:

MAX(sell_price)
40000.00

- **MIN() - Returns the smallest value**

Query:

select MIN(sell_price) from product_master;

Screenshot:

MIN(sell_price)
110.00

- **SUM() - Returns the sum**

Query:

select SUM(sell_price) from product_master;

Screenshot:

SUM(sell_price)
49110.00

POSTLAB QUESTIONS:

1.] Write a short note on DBA.

ANS: 1) The DBA is responsible for understanding and managing the overall database environment. By developing and implementing a strategic blueprint to follow when deploying databases within their organization, DBAs are instrumental to the ongoing efficiency of modern applications that rely on databases for data storage and access.

2) Without the DBA's oversight, it is inevitable that application and system outages, downtime and slowdowns will occur. Problems such as these result in business outages that can negatively affect revenue, customer experience and company reputation.

3.) The DBA is responsible for ensuring that databases and data are backed up appropriately and can be recovered correctly and quickly in the event of failure. Furthermore, the DBA ensures that databases are protected and secured. The DBA enacts measures to maintain that database integrity in terms of data accuracy and makes sure that unauthorized users cannot access the data.

4) Database administration is a vital component of the IT environment for any organization that relies on one or more database management systems

2.] Write different date functions and date formats.

ANS: Date Functions:

The following table lists the most important built-in date functions in MySQL.

FUNCTION	DESCRIPTION
NOW()	Returns the current date and time
CURDATE()	Returns the current date
CURTIME()	Returns the current time
DATE()	Extracts the date part of a date
EXTRACT()	Returns a single part of a date/time
DATE_ADD()	Adds a specified time interval to a date
DATE_SUB()	Subtracts a specified time interval from date
DATE_DIFF()	Returns the number of days btw two dates
DATE_FORMAT()	Displays date/time data in different formats

Date Formats:

We might need to format a date in different formats as per our requirements. We can use the SQL CONVERT() function in SQL server to formate Date Time in various formats. Syntax for the SQ: CONVERT() functions is:

SELECT CONVERT (data_type(length)), Date, DateFormat(code)

MySQL comes with the following data types for storing a date or a date/time value in the database:

- DATE - format YYYY-MM-DD
- DATETIME - format : YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format : YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

3.] Differentiate between group by and having with example.

ANS: 1. Having Clause:

Having clause is basically like the aggregate function with the GROUP BY clause. The HAVING clause is used instead of WHERE with aggregate functions. While the GROUP BY clause groups rows that have the same values into summary rows. The having clause is used with the where clause in order to find rows with certain conditions. The having clause is always used after the group by clause

```
SELECT COUNT (SALARIES) AS COUNT_SALARIES, EMPLOYEES
FROM EMPLOYEES
GROUP BY SALARIES
HAVING COUNT (SALARIES) >1;
```

2. Group By Clause:

The GROUP BY clause is often used with aggregate functions (MAX, SUM, AVG) to group the results by one or more columns or in simple words we can say that the GROUP BY clause is used in collaboration with the SELECT statement to arrange required data into groups. The GROUP BY statement groups rows that have the same values. This statement is used after the where clause. This statement is often used with some aggregate function like SUM, AVG, COUNT etc., to group the results by one or more columns.

```
SELECT COUNT (SALARIES) AS COUNT_SALARIES, EMPLOYEES
FROM EMPLOYEES
GROUP BY SALARIES;
```

HAVING CLAUSE	GROUP BY CLAUSE
1) It is used for applying some extra condition to the query.	1) The groupby clause is used to group the data according to particular column or row.
2) Having cannot be used without groupby clause.	2) Groupby can be used without having clause with the select statement.
3) The having clause can contain aggregate functions.	3) It cannot contain aggregate functions.
4) It restrict the query output by using some conditions.	4) It groups the output on basis of some rows or columns.

4.] Give different string functions.

ANS: String functions are used to perform an operation on input string and return an output string.

Followings are the string functions defined in SQL:

1) ASCII(): This function is used to find the ASCII value of a character.
2) CONCAT(): This function is used to add two words/strings.
3) FORMAT(): This function is used to display a number in the given format.
4) INSERT(): This function is used to insert the data into a database.
5) INSTR(): This function is used to find the occurence of an alphabet.
6) LCASE(): This function is used to convert the given string into lower case.
7) LENGTH(): This function is used to find the length of a word.
8) SPACE(): This function is used to write the given number of space
9) STRCMP(): This function is used to compare 2 strings.
10) REVERSE(): This function is used to reverse a string
11) REPLACE(): This function is used to cut the given string by removing the given sub string.
12) SUBSTR(): This function is used to find a sub string from the a string from a given posn.
13) TRIM(): This function is used to cut the given symbol from the string.
14) UCASE(): This function is used to make the string in upper case.
15) LOCATE(): This function is used to find the nth position of the given word in a string.