

SE-COMP(B)	Roll number :		
Experiment no. : 4	Date of Implementation :		
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	<p>SELECT: SELECT statement returns a result set of records from one or more tables.</p> <p>The select statement has optional clauses:</p> <ul style="list-style-type: none"> • 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. <p>Syntax:</p> <pre>SELECT<attribute list> FROM<table list> WHERE<condition></pre> <p>Where</p> <ul style="list-style-type: none"> • 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. <p>SQL Aggregate Functions</p> <p>SQL aggregate functions return a single value, calculated from values in a column.</p> <p>Useful aggregate functions:</p> <ul style="list-style-type: none"> • 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 <p>The SQL ORDER BY Keyword</p> <p>The ORDER BY keyword is used to sort the result-set by one or more columns.</p> <p>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.</p> <p>SQL ORDER BY Syntax</p> <pre>SELECT column_name, column_name FROM table_name ORDER BY column_name ASC DESC, column_name ASC DESC;</pre>

Procedure	<p>TASK 1:</p> <p>1. Create following table: Table name : sales_order</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Column Name</th><th style="text-align: center;">Data type</th><th style="text-align: center;">Size</th><th style="text-align: center;">Constraint</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">order_no</td><td style="text-align: center;">varchar</td><td style="text-align: center;">6</td><td style="text-align: center;">Primary Key</td></tr> <tr> <td style="text-align: center;">Order_date</td><td style="text-align: center;">date</td><td style="text-align: center;"></td><td style="text-align: center;">NOT NULL</td></tr> <tr> <td style="text-align: center;">Client_no</td><td style="text-align: center;">varchar</td><td style="text-align: center;">6</td><td style="text-align: center;">NOT NULL</td></tr> <tr> <td style="text-align: center;">Dely_addr</td><td style="text-align: center;">varchar</td><td style="text-align: center;">25</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Salesman_no</td><td style="text-align: center;">varchar</td><td style="text-align: center;">6</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Dely_type</td><td style="text-align: center;">char</td><td style="text-align: center;">1</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Billed_yn</td><td style="text-align: center;">char</td><td style="text-align: center;">1</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Dely_date</td><td style="text-align: center;">Date</td><td style="text-align: center;"></td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Order_status</td><td style="text-align: center;">varchar</td><td style="text-align: center;">10</td><td style="text-align: center;"></td></tr> </tbody> </table> <p>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.</p> <p>Task2: Use select with where statement with SQL aggregate functions for the tables created in Expt. no. 3</p>	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	
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																																							
Post Lab Questions:	<p>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.</p>																																								

Name: Brendan Lucas, Div: SE COMP B, Roll No: 8953

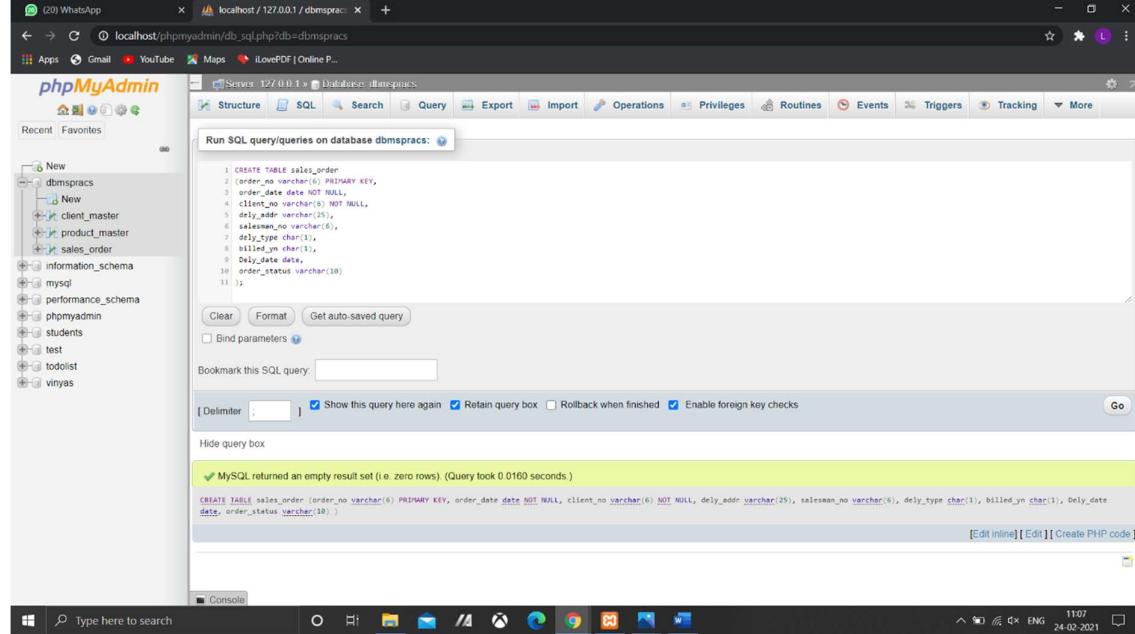
DBMS Practical Implementation, Lab 4

To Create Table:

```
CREATE TABLE sales_order
```

```
(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)
```

```
);
```



Inserting Values

```
INSERT INTO `sales_order` VALUES ('OD1071', '2021-02-12', 'CL1521', '1205, Rockdale, Bandstand',  
'SM0201', 'Y', 'Y', '2021-02-13', 'Shipped');
```

```
INSERT INTO `sales_order` VALUES ('OD1072', '2021-02-13', 'CL1521', '1205, Rockdale, Bandstand',  
'SM0201', 'Y', 'Y', '2021-02-15', 'Shipped');
```

```
INSERT INTO `sales_order` VALUES ('OD1073', '2021-02-10', 'CL1437', '1903, World towers',  
'SM0220', 'Y', 'Y', '2021-02-14', 'Shipped');
```

```
INSERT INTO `sales_order` VALUES ('OD1074', '2021-02-10', 'CL1077', '1501, Trump towers',  
'SM14022', 'Y', 'Y', '2021-02-17', 'Shipped');
```

```
INSERT INTO `sales_order` VALUES ('OD1075', '2021-01-13', 'CL1789', '612, Casa Rio', 'SM1452', 'Y',  
'Y', '2021-02-05', 'Shipped');
```

The screenshot shows the phpMyAdmin interface for the dbmspracs database. The left sidebar lists databases and tables, with 'sales_order' selected. The main area displays the contents of the 'sales_order' table. The table has the following columns: order_no, order_date, client_no, dely_addr, salesman_no, dely_type, billed_yn, Dely_date, and order_status. The data shows five rows of inserted values.

order_no	order_date	client_no	dely_addr	salesman_no	dely_type	billed_yn	Dely_date	order_status
OD1071	2021-02-12	CL1521	1205, Rockdale, Bandstand	SM0201	Y	Y	2021-02-13	Shipped
OD1072	2021-02-13	CL1521	1205, Rockdale, Bandstand	SM0201	Y	Y	2021-02-15	Shipped
OD1073	2021-02-10	CL1437	1903, World towers	SM0220	Y	Y	2021-02-14	Shipped
OD1074	2021-02-10	CL1077	1501, Trump towers	SM1402	Y	Y	2021-02-17	Shipped
OD1075	2021-01-13	CL1789	612, Casa Rio	SM1452	Y	Y	2021-02-05	Shipped

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

Query:- `SELECT * FROM `client_master` WHERE name LIKE "%_a%";`

The screenshot shows the phpMyAdmin interface for the 'client_master' table. The SQL query entered is `SELECT * FROM `client_master` WHERE name LIKE "%_a%"`. The results show four rows where the name starts with 'a'. The columns displayed are client_no, name, address, city, pincode, state, and Bal_due.

client_no	name	address	city	pincode	state	Bal_due
2005	Rajat Pramod	CMO, Jaipur	Jaipur	899358	Rajasthan	158261.00
3000	jayesh shah	CMO, Tripura	Tripura	950021	Mizoram	0.00
5051	ramdin shah	CMO, banglore	banglore	300021	Karnataka	75000.00

The screenshot shows the phpMyAdmin interface for the 'client_master' table. The SQL query entered is `SELECT * FROM `client_master` WHERE name LIKE "%_a%"`. The results show five rows where the name starts with 'a'. The columns displayed are client_no, name, address, city, pincode, state, and Bal_due.

client_no	name	address	city	pincode	state	Bal_due
2005	Rajat Pramod	CMO, Jaipur	Jaipur	899358	Rajasthan	158261.00
3000	jayesh shah	CMO, Tripura	Tripura	950021	Mizoram	0.00
5051	ramdin shah	CMO, banglore	banglore	300021	Karnataka	75000.00
9005	Jaspal Singh	CMO, Chandigarh	Chandigarh	515959	Punjab	38000.00
C005	kapil mishra	PMO, New Delhi	New Delhi	600021	New Delhi	1000.00

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

Query:- `SELECT * FROM `client_master` WHERE city LIKE "%a%";`

Screenshots:

The screenshot shows two instances of the phpMyAdmin interface. Both instances display the results of the same SQL query: `SELECT * FROM `client_master` WHERE city LIKE "%a%"`. The results are as follows:

client_no	name	address	city	pincode	state	Bal_due
2005	Rajat Pramod	CMO, Jaipur	Jaipur	808358	Rajasthan	158261.00
5051	ramadin shah	CMO, banglore	banglore	300021	Karnataka	75000.00

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

Query:- `SELECT * FROM `client_master` WHERE city="mumbai" ORDER BY name;`

Screenshots:

The screenshot shows the phpMyAdmin interface for the 'dbmspracs' database. The left sidebar lists databases and tables, with 'client_master' selected. The main area contains a SQL query editor with the following code:

```
SELECT * FROM `client_master` WHERE city="mumbai" ORDER BY name;
```

Below the query editor, the results are displayed in a table:

client_no	name	address	city	pincode	state	Bal_due
C010	mohan rajat	CMO, Mumbai	Mumbai	400001	Maharashtra	5000.00
C001	mohan sharma	CMO Pune	Mumbai	405001	Maharashtra	7500.00

This screenshot shows the same phpMyAdmin session after a refresh or a different operation. The results table now includes additional columns and rows:

client_no	name	address	city	pincode	state	Bal_due
C010	mohan rajat	CMO, Mumbai	Mumbai	400001	Maharashtra	5000.00
C001	mohan sharma	CMO Pune	Mumbai	405001	Maharashtra	7500.00
C002	mohan kumar	CMO, Mumbai	Mumbai	400001	Maharashtra	6000.00
C003	mohan kumar	CMO, Mumbai	Mumbai	400001	Maharashtra	5500.00

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

Query :- `SELECT * FROM `client_master` WHERE Bal_due>10000 ;`

Screenshots:

The screenshot shows the phpMyAdmin interface with the following details:

- Left Panel:** Shows the database structure with the 'client_master' table selected under the 'dbmspracs' database.
- Top Bar:** Shows the URL as `localhost/phpmyadmin/tbl_sql.php?db=dbmspracs&table=client_master`.
- SQL Query Box:** Contains the query `SELECT * FROM `client_master` WHERE Bal_due>10000 ;`.
- Results Area:** Displays the results of the query, showing two rows of data from the 'client_master' table where 'Bal_due' is greater than 10000.
- Table Headers:** The table has columns: client_no, name, address, city, pincode, state, and Bal_due.
- Data Rows:** The first row has values: 2005, Rajat Pramod, CMO, Jaipur, Jaipur, 699358, Rajasthan, 158261.00. The second row has values: 5051, ramadin shah, CMO, banglore, banglore, 300021, Karnataka, 75000.00.

The screenshot shows the phpMyAdmin interface with the following details:

- Left Panel:** Shows the database structure with the 'client_master' table selected under the 'dbmspracs' database.
- Top Bar:** Shows the URL as `localhost/phpmyadmin/tbl_sql.php?db=dbmspracs&table=client_master`.
- SQL Query Box:** Contains the query `SELECT * FROM `client_master` WHERE Bal_due>10000 ;`.
- Results Area:** Displays the results of the query, showing two rows of data from the 'client_master' table where 'Bal_due' is greater than 10000.
- Table Headers:** The table has columns: client_no, name, address, city, pincode, state, and Bal_due.
- Data Rows:** The first row has values: 2005, Rajat Pramod, CMO, Jaipur, Jaipur, 699358, Rajasthan, 158261.00. The second row has values: 5051, ramadin shah, CMO, banglore, banglore, 300021, Karnataka, 75000.00.
- Action Buttons:** Below the table, there are buttons for Edit, Copy, Delete, Check all, and With selected: Edit, Copy, Delete, Export.
- Bottom Buttons:** Includes Print, Copy to clipboard, Export, Display chart, and Create view.

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

Query:- SELECT * FROM `sales_order` WHERE order_date LIKE "%01%";

Screenshots:

The screenshot shows the phpMyAdmin interface for the dbmspracs database. The left sidebar lists tables including sales_order. The main area shows the results of the query: "SELECT * FROM `sales_order` WHERE order_date LIKE "%01%"". The results table has columns: order_no, order_date, client_no, dely_addr, salesman_no, dely_type, billed_yn, Dely_date, order_status. One row is displayed: order_no 01075, order_date 2021-01-13, client_no CL1789, dely_addr 612, Casa Rio, SM1452, salesman_no SM1452, dely_type Y, billed_yn Y, Dely_date 2021-02-05, order_status Shipped.

order_no	order_date	client_no	dely_addr	salesman_no	dely_type	billed_yn	Dely_date	order_status
01075	2021-01-13	CL1789	612, Casa Rio, SM1452	SM1452	Y	Y	2021-02-05	Shipped

8. Display the order information for client_no C001 and C002

Query :- SELECT * FROM sales_order WHERE client_no="C001" OR client_no="C002";

Screenshots:

The screenshot shows the phpMyAdmin interface for a database named 'dbmspracs'. The left sidebar lists various tables, including 'sales_order'. The main area displays the results of a SQL query: 'SELECT * FROM sales_order WHERE client_no="C001" OR client_no="C002"'. The results show two rows of data:

order_no	order_date	client_no	dely_addr	salesman_no	dely_type	billed_yn	Dely_date	order_status
OD1076	2021-02-10	C1001	Mannat, Mumbai	SM1078	Y	Y	2021-02-24	Shipped
OD1077	2021-02-15	C1002	Mannat, Mumbai	SM1020	Y	Y	2021-02-28	Shipped

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

Query :- SELECT * FROM product_master WHERE selling_price>2000 AND selling_price<=5000;

Screenshots:

The screenshot shows the phpMyAdmin interface with the following details:

- Database: dbmspracs
- Table: product_master
- SQL Query: `SELECT * FROM product_master WHERE selling_price>2000 AND selling_price<=5000;`
- Result Set:

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	selling_price
1001	Table Fan	5.25	1	500	2000	3000.00	4000.00
1003	Portable AC	9.50	1	500	200	2500.00	3000.00

The screenshot shows the phpMyAdmin interface with the following details:

- Database: dbmspracs
- Table: product_master
- SQL Query: `SELECT * FROM product_master WHERE selling_price>2000 AND selling_price<=5000;`
- Result Set:

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	selling_price
1001	Table Fan	5.25	1	500	2000	3000.00	4000.00
1003	Portable AC	9.50	1	500	200	2500.00	3000.00

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

Query:-

```
SELECT * FROM product_master WHERE selling_price>=1500;
```

```
UPDATE product_master SET selling_price=selling_price*1.5 WHERE selling_price>1500;
```

```
ALTER TABLE product_master CHANGE `selling_price` `new_price` decimal(8,2);
```

Screenshots:

Before:

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	selling_price
1001	Table Fan	5.25	1	500	2000	3000.00	4000.00
1002	Ceiling Fan	6.25	1	500	200	500.00	1000.00
1003	Portable AC	9.50	1	500	200	2500.00	3000.00
1007	Face Mask	5.00	2	500	1000	8.00	12.00

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	new_price
1001	Table Fan	5.25	1	500	2000	3000.00	4000.00
1002	Ceiling Fan	6.25	1	500	200	500.00	1000.00
1003	Portable AC	9.50	1	500	200	2500.00	3000.00
1007	Face Mask	5.00	2	500	1000	8.00	12.00

localhost/phpmyadmin/tbl_sql.php?db=dbmspracs&table=product_master

phpMyAdmin

Recent Favorites

- New
- dbmspracs
 - New
 - client_master
 - product_master
 - sales_order
- information_schema
- mysql
- performance_schema
- phpmyadmin
- students
- test
- todoist
- vinyas

Server: 127.0.0.1 > Database: dbmspracs > Table: product_master

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

SELECT * FROM product_master WHERE selling_price>1500

Show this query here again Retain query box Rollback when finished Enable foreign key checks Go

Hide query box

Showing rows 0 - 1 (total: 2) Query took 0.0010 seconds.

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	selling_price
1001	Table Fan	5.25	1	500	2000	3000.00	4000.00
1003	Portable AC	9.50	1	500	200	2500.00	3000.00

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Options Edit Copy Delete 1001 Edit Copy Delete 1003

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Query results operations

Print Copy to clipboard Export Display chart Create view

Type here to search

localhost/phpmyadmin/tbl_sql.php?db=dbmspracs&table=product_master

phpMyAdmin

Recent Favorites

- New
- dbmspracs
 - New
 - client_master
 - product_master
 - sales_order
- information_schema
- mysql
- performance_schema
- phpmyadmin
- students
- test
- todoist
- vinyas

Server: 127.0.0.1 > Database: dbmspracs > Table: product_master

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	selling_price
1001	Table Fan	5.25	1	500	2000	3000.00	4000.00
1003	Portable AC	9.50	1	500	200	2500.00	3000.00

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Options Edit Copy Delete 1001 Edit Copy Delete 1003

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query Label: Let every user access this bookmark

2 rows affected. (Query took 0.0028 seconds.)

UPDATE product_master SET selling_price=selling_price*1.5 WHERE selling_price<1500

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0047 seconds.)

ALTER TABLE product_master CHANGE `selling_price` `new_price` decimal(8,2)

Type here to search

localhost/phpmyadmin/sql.php?db=dbmspracs&table=product_master&pos=0

phpMyAdmin

Recent Favorites

- New
- dbmspracs
 - New
 - client_master
 - product_master
 - sales_order
- information_schema
- mysql
- performance_schema
- phpmyadmin
- students
- test
- todoist
- vinyas

Server: 127.0.0.1 > Database: dbmspracs > Table: product_master

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 3 (total: 4) Query took 0.00007 seconds.

SELECT * FROM `product_master`

Show this query here again Retain query box Rollback when finished Enable foreign key checks Go

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	new_price
1001	Table Fan	5.25	1	500	2000	3000.00	6000.00
1002	Ceiling Fan	6.25	1	500	200	500.00	1000.00
1003	Portable AC	9.50	1	500	200	2500.00	4500.00
1007	Face Mask	5.00	2	500	1000	8.00	12.00

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Options Edit Copy Delete 1001 Edit Copy Delete 1002 Edit Copy Delete 1003 Edit Copy Delete 1007

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key None

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query Label: Let every user access this bookmark

Type here to search

11. Count the total number of orders

Query:- `SELECT COUNT(order_no) FROM `sales_order`;`

Screenshots:

The screenshot shows the phpMyAdmin interface on a Windows desktop. The left sidebar lists databases (dbmspracs, dbmspracs, client_master, product_master, sales_order, information_schema, mysql, performance_schema, phpmyadmin, students, test, todolist, vinyas). The main area shows a query editor with the following content:

```
1 SELECT COUNT(order_no) FROM `sales_order`;
```

Below the query, the results are displayed:

order_no	order_date	client_no	delv_addr	salesman_no	delv_type	billed_yn	Delay_date	order_status

At the bottom, a message indicates the query was executed successfully:

Your SQL query has been executed successfully.

SELECT COUNT(order_no) FROM `sales_order`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Query results operations: Print, Copy to clipboard, Export, Display chart, Create view.

Bookmark this SQL query: Label: [] Let every user access this bookmark.

12. Calculate the average price of all the product

Query:-

```
SELECT AVG(new_price) FROM `product_master`;
```

```
SELECT * FROM product_master;
```

Screenshots:

The screenshot shows two instances of the phpMyAdmin interface. Both instances are connected to the database 'dbmspracs' and the table 'product_master'. In the top instance, the SQL query `SELECT AVG(new_price) FROM 'product_master'` is run, resulting in a single row output: `AVG(new_price)` with the value `2878.000000`. The bottom instance shows the result of the query `SELECT * FROM product_master`, which displays all columns and rows from the table.

13. Determine minimum and maximum product prices

Query:-

```
SELECT MIN(new_price) AS Smallest_Price FROM `product_master`;
```

```
SELECT MAX(new_price) AS Largest_Price FROM `product_master`;
```

Screenshots:

The image contains two side-by-side screenshots of the phpMyAdmin database management tool running on a Windows operating system.

Screenshot 1 (Left): This screenshot shows the SQL query editor window. The left sidebar lists databases and tables, with 'product_master' selected. The main area contains the following SQL code:

```
1: SELECT MIN(new_price) AS Smallest_Price FROM `product_master`;
2: SELECT MAX(new_price) AS Largest_Price FROM `product_master`;
```

The results pane shows the output of the first query:

```
Smallest_Price
+-----+
| 10.00 |
+-----+
```

Screenshot 2 (Right): This screenshot shows the results of the second query from the previous screenshot. The results pane displays:

```
Largest_Price
+-----+
| 6000.00 |
+-----+
```

Both screenshots show the standard phpMyAdmin interface with toolbars, navigation menus, and status bars indicating the date and time (26-02-2021, 11:34).

DBMS_EXPT4_SEcomp8.docx - C:\xampp\tmp\phpmyadmin\localhost\phpmyadmin\tbl_sql.php?db=dbmspracs&table=product_master

localhost/phpmyadmin/tbl_sql.php?db=dbmspracs&table=product_master

SQL MIN() and MAX() Functions

(21) WhatsApp

phpMyAdmin

Recent Favorites

New dbmspracs New client_master product_master sales_order information_schema mysql performance_schema phpmyadmin students test todolist vinyas

Showing rows 0 - 0 (total, Query took 0.0028 seconds.)

SELECT MIN(new_price) AS Smallest_Price FROM `product_master`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table

+ Options Smallest_Price 12.00

Show all Number of rows: 25 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query Label: Let every user access this bookmark

Bookmark this SQL query

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Console

Type here to search

11:34 26-02-2021

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

Query:-

```
SELECT COUNT(new_price) FROM `product_master` WHERE new_price>1500;
```

```
SELECT * FROM `product_master` WHERE new_price>1500;
```

Screenshots:

The screenshot shows the phpMyAdmin interface for the 'product_master' table. The left sidebar lists databases and tables, with 'product_master' selected. The main area contains two queries:

```
1: SELECT COUNT(new_price) FROM `product_master` WHERE new_price>1500;
2: SELECT * FROM `product_master` WHERE new_price>1500;
```

The results pane shows the output of the first query:

```
Count(new_price)
```

A message at the bottom indicates the query was executed successfully.

The screenshot shows the phpMyAdmin interface for the 'product_master' table. The left sidebar lists databases and tables, with 'product_master' selected. The main area displays the results of the query:

```
SELECT * FROM `product_master` WHERE new_price>1500;
```

The results table shows two rows of data:

product_no	description	profit_percent	unit_measure	qty_on_hand	reorder_level	cost_price	new_price
1001	Table Fan	5.25	1	500	2000	3000.00	6000.00
1003	Portable AC	9.50	1	500	200	2500.00	4500.00

Below the table, there are options for 'Query results operations' such as Print, Copy to clipboard, Export, Display chart, and Create view.

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

Query:- `SELECT order_no,order_date FROM `sales_order` ORDER BY order_date;`

Screenshots:

The screenshot shows two instances of the phpMyAdmin interface. Both instances are connected to the 'dbmspracs' database and the 'sales_order' table. The top instance shows the SQL query `SELECT order_no,order_date FROM `sales_order` ORDER BY order_date;` in the query editor. The results pane displays the following data:

order_no	order_date
OD1075	2021-01-13
OD1073	2021-02-10
OD1074	2021-02-10
OD1076	2021-02-10
OD1071	2021-02-12
OD1072	2021-02-13
OD1077	2021-02-15

The bottom instance of phpMyAdmin shows the same results, indicating that the query has been executed again.

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

Query:-

```
SELECT order_no,DATE_FORMAT(order_date,"%d/%M/%Y") FROM sales_order ORDER BY order_date;
```

Screenshots

The screenshot shows the phpMyAdmin interface for a MySQL database named 'dbmspracs'. The 'sales_order' table is selected. In the SQL tab, the following query is entered:

```
SELECT order_no,DATE_FORMAT(order_date,"%d/%M/%Y") FROM sales_order ORDER BY order_date;
```

The results show 6 rows of data, each with an order number and its corresponding date formatted as 'dd-Month-yy'. The columns listed are order_no, order_date, client_no, delv_addr, salesmen_no, delv_type, billed_yn, Delv_date, and order_status.

order_no	order_date	client_no	delv_addr	salesmen_no	delv_type	billed_yn	Delv_date	order_status
OD1075	13/January/2021							
OD1073	10/February/2021							
OD1074	10/February/2021							
OD1076	10/February/2021							
OD1071	12/February/2021							
OD1072	13/February/2021							
OD1077	15/February/2021							

The screenshot shows the phpMyAdmin interface after the query has been executed. The results are displayed in a grid format. The 'order_no' column is sorted in ascending order. The 'order_date' column is also visible, showing the dates in their original format. The 'DATE_FORMAT(order_date,"%d/%M/%Y")' column shows the dates reformatted as 'dd-Month-yy'.

order_no	order_date	DATE_FORMAT(order_date,"%d/%M/%Y")
OD1075	13/January/2021	13-Jan-21
OD1073	10/February/2021	10-Feb-21
OD1074	10/February/2021	10-Feb-21
OD1076	10/February/2021	10-Feb-21
OD1071	12/February/2021	12-Feb-21
OD1072	13/February/2021	13-Feb-21
OD1077	15/February/2021	15-Feb-21

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

Query:- `SELECT order_no,DATE_FORMAT(Dely_date,"%M %d %Y") FROM sales_order ORDER BY Dely_date;`

Screenshots:

The screenshot shows two instances of the phpMyAdmin interface. The top instance displays the results of the executed SQL query:

```
1 SELECT order_no,DATE_FORMAT(Dely_date,"%M %d %Y") FROM sales_order ORDER BY Dely_date;
```

The results table shows the following data:

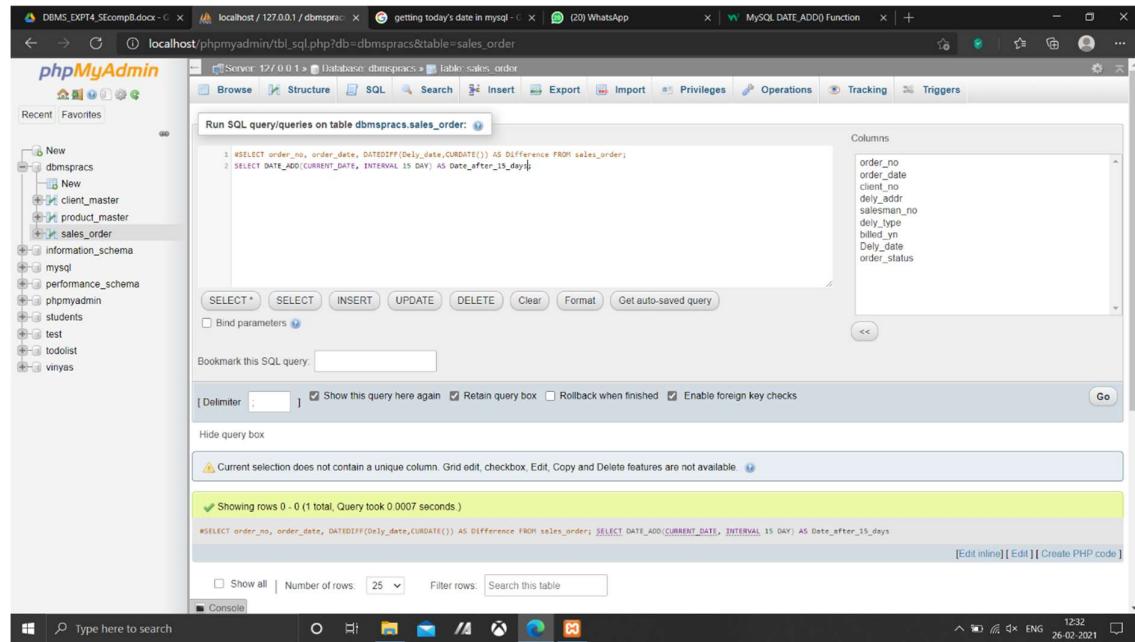
order_no	DATE_FORMAT(Dely_date,"%M %d %Y")
OD1075	February 05 2021
OD1071	February 13 2021
OD1073	February 14 2021
OD1072	February 15 2021
OD1074	February 17 2021
OD1076	February 24 2021
OD1077	February 26 2021

The bottom instance of phpMyAdmin shows the same table structure and data, but with the results table currently empty.

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

Query:- `SELECT DATE_ADD(CURRENT_DATE, INTERVAL 15 DAY) AS Date_after_15_days;`

Screenshots:



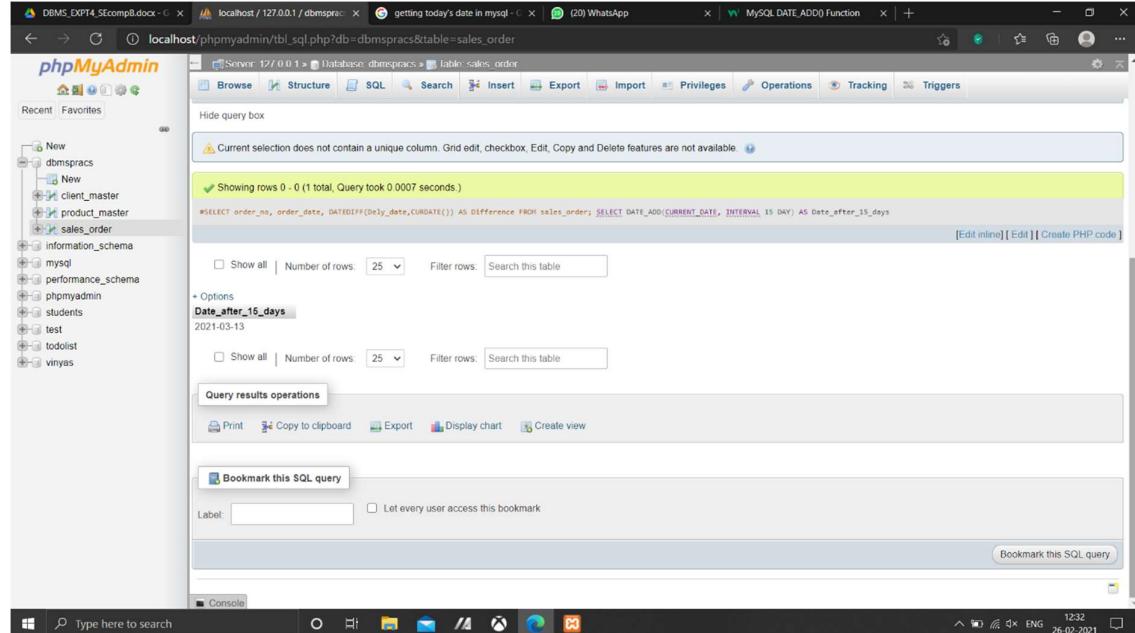
The screenshot shows the phpMyAdmin interface for the dbmspracs database. In the left sidebar, the 'sales_order' table is selected. The main area contains the following SQL query:

```
1 SELECT order_no, order_date, DATEDIFF(DATE(), date,CURDATE()) AS Difference FROM sales_order;
2 SELECT DATE_ADD(CURRENT_DATE, INTERVAL 15 DAY) AS Date_after_15_days;
```

The results pane shows the output of the query:

```
+-----+-----+
| order_no | Date_after_15_days |
+-----+-----+
|        | 2021-03-13          |
+-----+-----+
```

Below the results, there is a note: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." The status bar at the bottom right indicates "1232 26-02-2021".



The screenshot shows the phpMyAdmin interface for the dbmspracs database. In the left sidebar, the 'sales_order' table is selected. A new view named 'Date_after_15_days' is being created with the following SQL query:

```
#SELECT order_no, order_date, DATEDIFF(DATE(), date,CURDATE()) AS Difference FROM sales_order; SELECT DATE_ADD(CURRENT_DATE, INTERVAL 15 DAY) AS Date_after_15_days
```

The 'Label' field is empty, and the 'Let every user access this bookmark' checkbox is unchecked. The status bar at the bottom right indicates "1232 26-02-2021".

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

Query:- `SELECT order_no, order_date, DATEDIFF(Dely_date,CURDATE()) AS Difference FROM sales_order;`

Screenshots:

The screenshot shows two instances of the phpMyAdmin interface. In the top instance, a SQL query is run on the 'sales_order' table:

```
1 SELECT order_no, order_date, DATEDIFF(Dely_date,CURDATE()) AS Difference FROM sales_order;
```

The results show the following data:

order_no	order_date	Difference
OD1071	2021-02-12	-13
OD1072	2021-02-13	-11
OD1073	2021-02-10	-12
OD1074	2021-02-10	-9
OD1075	2021-01-13	-21
OD1076	2021-02-10	-2
OD1077	2021-02-15	2

In the bottom instance, the results are displayed in a grid format:

order_no	order_date	Difference
OD1071	2021-02-12	-13
OD1072	2021-02-13	-11
OD1073	2021-02-10	-12
OD1074	2021-02-10	-9
OD1075	2021-01-13	-21
OD1076	2021-02-10	-2
OD1077	2021-02-15	2

(i) Write a short note on DBA.

Ans) Database Administrator manages and controls three levels of database like internal level, conceptual level, and external level of Database management system architecture and in discussion with comprehensive user community gives definition of world view of database. It then provides external view of different users and applications.

(ii) Database Administrator ensures held responsible to maintain integrity and security of database restricting of unauthorized users. It grants permission to users of database and contains profile of each and every user in database.

(iii) Database Administrator also held accountable that database is protected and secured and that any chance of database keeps at minimum.

Q2 Write different date functions & Date formats

Ans

Date Functions:

- (i) Date functions are used to manipulate date and data in date format.
- (ii) They may or may not take date formats as arguments.
- (iii) They come built-in with MySQL and can be used from finding current date and time to converting to dates till finding difference between days.

e.g:- DATE(DIFF), Date_Format(), Date_ADD(), etc

Date Formats:-

- (i) Date formats are used to specify the format of the date to be returned.
 - (ii) The default format for MySQL is 'yyyy-mm-dd'. In this format data is stored in the database.
 - (iii) They are passed as arguments within a function call. They are usually enclosed by a percent sign (%) proceeded by an alphabet.
- e.g.: %d, %Y, %m, %a, etc.

Q3 Differentiate between the group by & having clauses

Ans The group by clause:-

(i) The group by clause is used to group the data according to particular column or row.

(ii) Groupby can be used without having clause only with the select statement.

(iii) It cannot contain aggregate functions

(iv) It groups the output on bases of rows & columns. Some

The Having Clause:-

(i) It is used for applying some extra conditions to the query

(ii) Having cannot be used without groupby clause

(iii) The having clause can contain aggregate functions

(iv) It restricts the query output by using some condition

Q4 Give the name of some string functions
 Ans

(i) ASCII()

Returns the ASCII value of the specific character.

(ii) CHAR_LENGTH()

Returns the length of a string (In characters)

(iii) CONCAT()

Adds two or more expressions together

(iv) FIND_IN_SET()

Returns the position of a string within a list of strings

(v) INSERT()

Inserts a string within a string at the specified position and for a certain number of characters.

(vi) LCASE();

Converts a string to lower-case.

(vii) LENGTH();

Returns the length of a string (in bytes)

(viii) Reverse();

Reverses a string and ~~not~~ returns the result;

(ix) STRCMP();

Compares two strings