

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

## **Institute of Computer Technology**

### **B. Tech Computer Science and Engineering**

**Sub: Database Management System (2CSE301)**

#### **Practical 3: Performing queries with various operators and functions.**

Scenario : Mohan was worried about total income to getting raised every month for which he has to query differently in sales table. Thus he suggested IT Company to have multiple options to see the count of sales happen every day or weekly or monthly.

#### **COMPUTATION ON TABLE DATA**

None of the techniques used till now allows displays of some data from a table after some arithmetic has been done with it.

Arithmetic and logical operators give a new dimension to SQL sentences.

#### **Arithmetic Operators:**

Oracle allows arithmetic operator to be used while viewing records from a table or while performing Data Manipulation operations such as Insert, Update and Delete.

+	Addition	*	Multiplication
-	Subtraction	**	Exponentiation
/	Division	( )	Enclosed operation

For example:

Retrieve the content of the column p\_no, description and compute 5% of the values contained in the column sell\_price for each row from the table product\_master.

```
Sql>SELECT p_no, description, sell_price*0.05
```

```
FROM product_master;
```

#### **Renaming Columns used with Expression Lists:**

When displaying the result of a query, SQL \*PLUS normally uses the selected column's name as the column heading.

These column names may however be short and cryptic; they can be changed for better understanding of the query result by entering an alias, or substitute name, after the column name in the select clause.

```
Sql>SELECT columnname result_columnname, columnname result_columnname
```

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

**FROM** table name For

example:

Retrieve the content of the column p\_no, description and compute 5% of the values contained in the column sell\_price for each row from the table product\_master. Rename sell\_price \* 0.05 as **Increase**.

```
Sql>SELECT p_no, description, sell_price*0.05 Increase
      FROM product_master;
```

### **Logical Operators:**

Logical operators that can be used in SQL sentence are:

#### **1. AND operator:**

The Oracle engine will process all rows in a table and display the result only when all of the conditions specified using the AND operator are satisfied.

```
sql >SELECT column list
      FROM tablename
      WHERE columnname AND columnname;
```

```
Sql>SELECT p_no, desc, p_percent
      FROM product_master
      WHERE p_percent>=10 AND p_percent<=20;
```

#### **2. OR operator:**

The Oracle engine will process all rows in a table and display the result only when any of the conditions specified using the OR operators are satisfied.

```
sql >SELECT column list
      FROM tablename
      WHERE columnname OR columnname;
```

```
sql >SELECT c_no, name, address, pincode
      FROM client_master
      WHERE (pincode=400125 OR pincode=400126);
```

### **3. NOT operator:**

The Oracle engine will process all rows in a table and display the result only when none of the conditions specified using the NOT operator are satisfied.

```
Sql> SELECT c_no, name, address, pincode
      FROM client_master WHERE NOT (city='Bombay' or city='Delhi');
```

### **Range Searching:**

In order to select data that is within a range of values, the **BETWEEN** operator is used. This operator allows the selection of rows that contain values within a specified lower and upper limit.

```
sql >SELECT column list from tablename
      WHERE column BETWEEN min _value AND max_value;
```

```
sql >SELECT c_no, name, address, pincode
      FROM client_master
      WHERE bal_due BETWEEN 100 AND 500;
```

### **Note:**

.BETWEEN is an inclusive operator i.e. if either the min value or the max value is found, as well as any in between, the row is returned.

### **4. NOT BETWEEN**

Rows not having value in the range specified, and also not having value equal; to min or the max value is returned.

```
sql >SELECT column list from tablename
      WHERE column NOT BETWEEN min _value AND max_value;
sql >SELECT c_no, name, address, pincode
      FROM client_master
      WHERE bal_due NOT BETWEEN 100 AND 500;
```

### **Pattern Matching:**

#### **1. LIKE**

## **Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

Allows comparison of one string value with another string value, which is not identical. This is achieved by using wildcard characters. Two wildcard characters that are available are:

- The percent sign (%) that matches any string.
- The Underscore ( \_ ) that matches any single character.

```
sql >SELECT column list FROM tablename  
      WHERE column LIKE 'pattern';
```

OR

```
      WHERE column NOT LIKE 'pattern';
```

For example:

Retrieve all information about suppliers whose name begin with the letter 'ja' from supplier\_master.

```
sql >SELECT * FROM supplier_master  
      WHERE s_name LIKE 'ja%';
```

### **2. IN**

This operator can be used to select rows that match one of the values included in the list.

```
sql>SELECT columnlist FROM tablename  
      WHERE columnlist IN (list of values);
```

For example:

Retrieve the details from supplier table where supplier name is either Aman or Vimal or Ajay.

```
sql>SELECT s_no, name, city, address, pincode  
      FROM supplier_master  
      WHERE name IN ('Aman', 'Vimal', 'Ajay');
```

### **3. NOT IN**

The **NOT IN** predicate is the opposite of the IN predicate. This will select all the rows where values do not match all of the values in the list.

```
sql>SELECT columnlist FROM tablename  
      WHERE columnlist NOT IN (list of values);
```

### **4. IS NULL**

This operator is used to compare the value in the column with NULL and return the row accordingly.

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

sql >**SELECT** column list **FROM** tablename

**WHERE** column is **NULL**;

OR

**WHERE** column is not **NULL**;

### **ORACLE FUNCTIONS:**

Oracle functions serve the purpose of manipulating data items and returning result. Functions are also capable of accepting user-supplied variables or constants and operating on them. Such variables or constants are called as argument. Any number of arguments can be passed to a function in the following format:

**Function\_name (argument1, argument2, ...).**

Oracle functions can be clubbed together depending upon whether they operate on a single row or a group of rows retrieved from a table. Accordingly, functions can be classified as follows:

#### **Group Functions (Aggregate Function):**

Functions that act on a set of values are called as group functions. For example, SUM, is a function which calculates the total of a set of numbers. A group function returns a single result row a group of queried rows.

#### **Scalar Function (Single Row Function):**

Functions that act on only one value at a time are called as scalar functions. For example, LENGTH, is a function, which calculates the length of one particular string value. A single row function returns one result for every row of a queried table or view.

Single row function can be further grouped together by the data type of their arguments and return values. For example, LENGTH, relates to the string Data type. Functions can be classified corresponding to different data types as:

String functions                   : Work for String Data type

Numeric functions               : Work for Number Data type

Conversion functions           : Work for Conversion of one type to another.

Date functions                 : Work for Date Data type

### **Aggregate Functions:**

<b>AVG</b>	Syntax	AVG([DISTINCT ALL]n)
	Purpose	Return average value of n ignoring null values.

	Example	Select AVG(sell_price) “Average” from p_master;
	Output	<u>Average</u> 2012.3654
<b>MIN</b>	Syntax	MIN([DISTINCT ALL]expr)
	Purpose	Return minimum value of ‘expr’.
	Example	Select MIN(bal_due) “Min_bal” from c_master;
	Output	<u>Min_bal</u> 0
<b>COUNT</b>	Syntax	MIN([DISTINCT ALL]expr)
	Purpose	Return the number of rows WHERE ‘expr’ is not null .
	Example	Select COUNT(p_no) “Products” from P_master;
	Output	<u>Products</u> 9
<b>COUNT(*)</b>	Syntax	COUNT(*)
	Purpose	Return the number of rows in the table, including duplicates and those with nulls..
	Example	Select COUNT(*) “Total” from C_master;
	Output	<u>Total</u> 9
<b>MAX</b>	Syntax	MAX([DISTINCT ALL]expr)
	Purpose	Return maximum value of ‘expr’.
	Example	Select MAX(bal_due) “Maximum” from c_master;
	Output	<u>Maximum</u> 15000
<b>SUM</b>	Syntax	SUM([DISTINCT ALL]n)
	Purpose	Return Sum of values of ‘n’.
	Example	Select SUM(bal_due) “Balance” from c_master;
	Output	<u>Balance</u> 22000

### **Numeric Functions:**

<b>ABS</b>	Syntax	ABS(n)
	Purpose	Return the absolute values of ‘n’.

	Example	Select ABS(-15) “Absolute” from dual;
	Output	<u>Absolute</u> 15
<b>POWER</b>	Syntax	POWER(m,n)
	Purpose	Returns m raised to nth power. N must be an integer, else an error is returned.
	Example	Select POWER(3,2) “Raised” from dual;
	Output	<u>Raised</u> 9
<b>ROUND</b>	Syntax	ABS(n[,M])
	Purpose	Returns ‘n’ rounded to ‘m’ places right the decimal point. If ‘m’ is omitted ‘n’ is rounded to 0 places. ‘m’ can be negative to round off digit left of the decimal point ‘m’ must be an integer.
	Example	Select ROUND(15.19,1) “Round” from dual;
	Output	<u>Round</u> 15.2
<b>SQRT</b>	Syntax	SQRT(n)
	Purpose	Returns square root of ‘n’. if n<0, NULL. SQRT returns a real result.
	Example	Select SQRT(25) “Square root” from dual;
	Output	<u>Square root</u> 5

### **String Functions:**

<b>LOWER</b>	Syntax	LOWER(char)
	Purpose	Return char, with all letters in lowercase.
	Example	Select LOWER(‘XYZ’) “Lower” from dual;
	Output	<u>Lower</u> xyz
<b>INITCAP</b>	Syntax	INITCAP(char)
	Purpose	Return STRING with first letter in upper case.
	Example	Select INITCAP(‘COMP DEPT’) “Title Case” from dual;
	Output	<u>Title Case</u> Comp Dept
<b>UPPER</b>	Syntax	UPPER(char)
	Purpose	Return char, with all letters in uppercase.

	Example	Select UPPER('xyz') "Upper" from dual;
	Output	<u>Upper</u> XYZ
<b>SUBSTR</b>	Syntax	UPPER(char, M[,n])
	Purpose	Return a portion of char, beginning at character 'm' exceeding up to 'n' characters. If 'n' is omitted, result is returned up to the end char. The first position of char is 1.
	Example	Select SUBSTR('SECURE',3,4) "Substring" from dual;
	Output	<u>Substring</u> CURE
<b>LENGTH</b>	Syntax	LENGTH(char)
	Purpose	Return the length of character.
	Example	Select LENGTH('xyz') "Length" from dual;
	Output	<u>Length</u> 3
<b>LTRIM</b>	Syntax	LTRIM(char[,Set])
	Purpose	Return characters from the left of char with initial.
	Example	Select LTRIM('College','C') "Left" from dual;
	Output	<u>Left</u> ollege
<b>RTRIM</b>	Syntax	RTRIM(char[,Set])
	Purpose	Return char, with final characters removed after the last character not I the set. 'set' is optional, it defaults to spaces.
	Example	Select RTRIM('College','e') "Right" from dual;
	Output	<u>Right</u> Colleg
<b>LPAD</b>	Syntax	LPAD(char1,n[,char2])
	Purpose	Return 'char1', left padded to length 'n' with the sequence of characters in 'char2', 'char2, defaults to blanks.
	Example	Select LPAD('Page 1',10,'*') "Lpad" from dual;
	Output	<u>Lpad</u> ****Page 1
<b>RPAD</b>	Syntax	RPAD(char1,n[,char2])



	Purpose	Return 'char1', right- padded to length 'n' with the characters in 'char2', replicated as many times as necessary. If 'char2' is omitted, right-pad is with blanks.
	Example	Select RPAD('page',10,'x') "Rpad" from dual;
	Output	<u>Rpad</u> Pagexxxxxx

## **Conversion Functions:**

<b>TO_NUMBER</b>	Syntax	TO_NUMBER(char)
	Purpose	Converts 'char' , a character value containing a number to a value of number datatype.
	Example	Update P_master set sell_price= sell_price + TO_NUMBER(SUBSTR('\$100',2,3));  Here the value 100 will be added to every products selling price in the product_master table.
<b>TO_CHAR</b>	Syntax	TO_CHAR(n[,fmt])
	Purpose	Converts a value of number data type to a value of char data type, using the optional format string. It accepts a number (n) and a numeric format (fmt) in which the number has to appear. If 'fmt' is omitted, 'n' is converted to a char value exactly long enough to hold significant digits.
	Example	Select TO_CHAR(17145,'\$099,999') "char" from dual;
	Output	<u>Char</u> \$017,145
<b>TO_CHAR</b>	Syntax	TO_CHAR(date[,fmt])
	Purpose	Converts a value of DATE data type to a value of char data type, using the optional format string. It accepts a date (date), as well as format (fmt) in which the date has to appear. 'fmt' must be a date format.. If 'fmt' is omitted, 'date' is converted to a char value in the default date format, i.e. "DD-MON-YY".
	Example	Select TO_CHAR(O_DATE,'Month DD, YYYY') "Format" from s_order where o_no='o42541';
	Output	<u>Format</u> January 26, 20006



## **Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

When an arithmetic exercise is to be performed such as 2\*2 or 4/3 etc., there really is no table being referenced; only numeric literals are being used.

To facilitate such calculation via a SELECT, Oracle provides a dummy table called DUAL, against which SELECT statements that are required to manipulate numeric literals can be fired, and output obtained.

Sql>**SELECT 2\*2 FROM DUAL;**

Output:

2*2
4

### **SYSDATE:**

Sysdate is a pseudo column that contains the current date and time. It requires no arguments when selected from the table DUAL and returns the current date.

Sql>**SELECT sysdate FROM DUAL;**

Output:

Sysdate
06-jun-06

### **Exercise:**

Create table sales\_order and insert data as given below.

Column Name	Data Type	Size
Order_no	Varchar	6
Order_date	Date	
Client_no	Varchar	6
S_no	Varchar	6
Dely_type	Char	1
Billed_yn	Char	1
Dely_date	Date	
Order_status	Varchar	10

Order_no	Order_date	Client_no	S_no	Dely_type	Billed_yn	Dely_date	Order_status
----------	------------	-----------	------	-----------	-----------	-----------	--------------

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

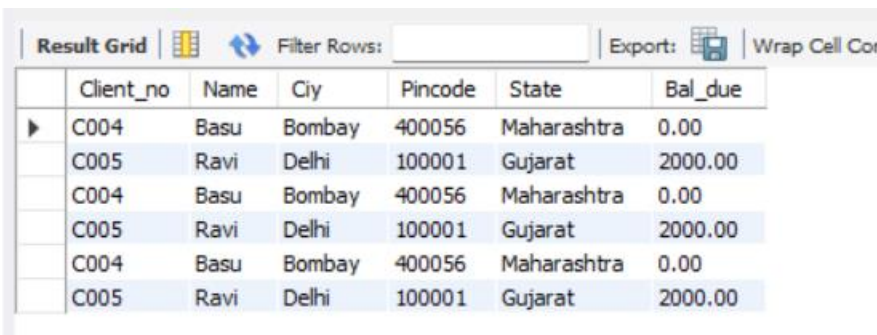
O1901	06/12/2015	C001	S001	F	N	06/20/2015	InProcess
O1902	01/25/2015	C002	S002	P	N	06/27/2015	Cancelled
O4665	02/18/2015	C003	S003	F	Y	02/20/2015	Fullfilled
O1903	04/03/2015	C001	S001	F	Y	04/07/2015	Fullfilled
O4666	05/20/2015	C004	S002	P	N	05/22/2015	Cancelled
O1908	05/24/2015	C005	S003	F	N	05/26/2015	InProcess

Solve the following queries using the database given in practical 1 and above table.

**Queries on computation on table data:**

1) Find the name of all clients having 'a' as the second letter in their names

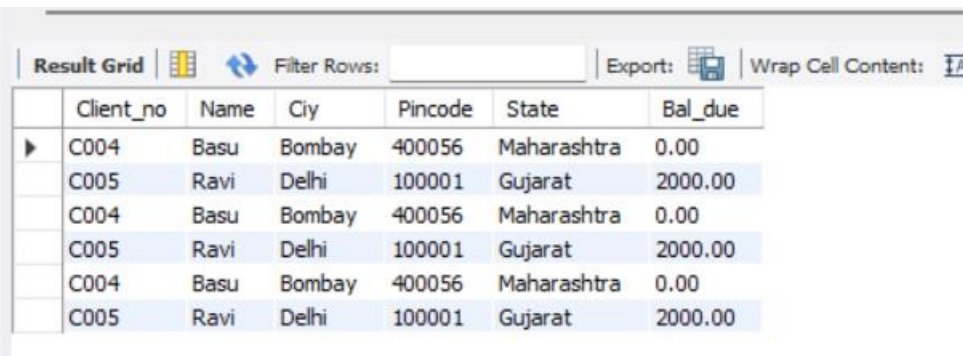
select \* from client\_master where name like '\_a%';



	Client_no	Name	Ciy	Pincode	State	Bal_due
▶	C004	Basu	Bombay	400056	Maharashtra	0.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00
	C004	Basu	Bombay	400056	Maharashtra	0.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00
	C004	Basu	Bombay	400056	Maharashtra	0.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00

2) Find out the clients whose name is four character long and second letter is 'a'.

select \* from client\_master where name like '\_a%' and length(name)=4;



	Client_no	Name	Ciy	Pincode	State	Bal_due
▶	C004	Basu	Bombay	400056	Maharashtra	0.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00
	C004	Basu	Bombay	400056	Maharashtra	0.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00
	C004	Basu	Bombay	400056	Maharashtra	0.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00

3) Find out the name of city whose second last character is 'a'.

select \* from client\_master where City like '%a\_';

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

Result Grid

Filter Rows:

Export:

Wrap

	Client_no	Name	Ciy	Pincode	State	Bal_due
▶	C001	Ivan	Bombay	400054	Maharashtra	1000.00
	C003	Pramada	Bombay	400057	Maharashtra	5000.00
	C004	Basu	Bombay	400056	Maharashtra	0.00
	C006	Rukmani	Bombay	400050	Maharashtra	0.00

4) Print the list of clients whose bal\_due is greater than or equal to 10000.

select \* from client\_master where Bal\_due >= 10000;

Result Grid

Filter Rows:

Export:

Wrap

	Client_no	Name	Ciy	Pincode	State	Bal_due
--	-----------	------	-----	---------	-------	---------

5) Print the information from sales\_order table for orders placed in the month of January.

select \* from Sales\_order where month(Order\_date)=01;

Client_no	Name	Ciy	Pincode	State	Bal_due
C001	Ivan	Bombay	400054	Maharashtra	1000.00
C003	Pramada	Bombay	400057	Maharashtra	5000.00
C004	Basu	Bombay	400056	Maharashtra	0.00
C005	Ravi	Delhi	100001	Gujarat	2000.00
C006	Rukmani	Bombay	400050	Maharashtra	0.00

6) Display the order information for client\_no 'C003' and 'C001'.

select \* from client\_master where Client\_no='C003' OR Client\_no='C001';

Result Grid

Filter Rows:



Export:

Wrap Ce

	Client_no	Name	Ciy	Pincode	State	Bal_due
▶	C001	Ivan	Bombay	400054	Maharashtra	1000.00
	C003	Pramada	Bombay	400057	Maharashtra	5000.00

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

select description, sell\_price, sell\_price\*0.15 as new\_price from product\_master where Sell\_price > 2000 AND Sell\_price <= 5000;

Result Grid			Filter Rows:
	description	sell_price	new_price
	Keyboards	3150.00	472.5000

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

- 8) Find products whose selling price is more than 1500. Calculate a new selling price as, original selling price \* .15. Rename the new column in the above query as new\_price.

```
select description, sell_price, sell_price * 0.15 as new_price from product_master where Sell_price>5000;
```

	description	sell_price	new_price
▶	Monitor	12000.00	1800.0000
	Cd Drive	5250.00	787.5000
	1.44 Drive	8400.00	1260.0000

- 9) List the names, city and state of clients who are not in the state of 'Maharashtra'.

```
select * from client_master where not State='Maharashtra';
```

	Client_no	Name	Ciy	Pincode	State	Bal_due
▶	C005	Ravi	Delhi	100001	Gujarat	2000.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00
	C005	Ravi	Delhi	100001	Gujarat	2000.00

- 10) Count the total number of orders.

```
select count(Order_no) from sales_order;
```

	count(Order_no)
▶	0

- 11) Calculate the average price of all products.

```
select avg(sell_price),avg(Cost_price) from product_master;
```

	avg(sell_price)	avg(Cost_price)
▶	5970.000000	5686.000000

- 12) Determine the maximum and minimum product prices. Rename the output as max\_price and min\_price respectively.

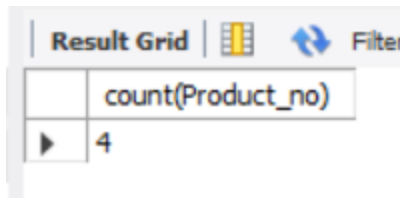
```
select max(sell_price),min(sell_price),max(sell_price) as max_price,min(sell_price) as min_price from product_master;
```

	max(sell_price)	min(sell_price)	max_price	min_price
▶	12000.00	1050.00	12000.00	1050.00

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

13) Count the number of products having price greater than or equal to 1500.

```
select count(Product_no) from product_master where Sell_price >= 1500;
```

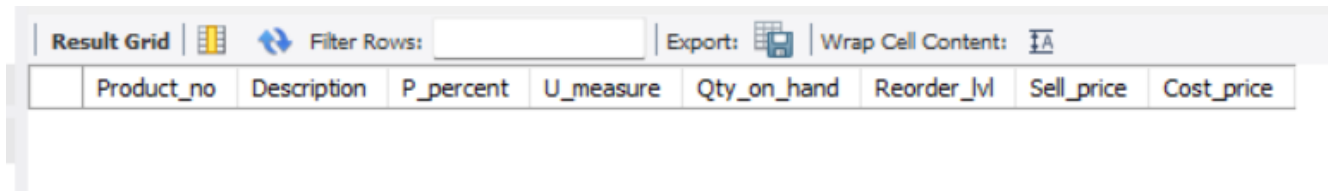


The screenshot shows a 'Result Grid' with a single row containing the value '4' under the column header 'count(Product\_no)'.

count(Product_no)
4

14) Find all the products whose qty\_on\_hand is less than reorder level.

```
select * from product_master where Qty_on_hand < Reorder_lvl;
```



The screenshot shows a 'Result Grid' with columns: Product\_no, Description, P\_percent, U\_measure, Qty\_on\_hand, Reorder\_lvl, Sell\_price, and Cost\_price. The grid is currently empty.

Product_no	Description	P_percent	U_measure	Qty_on_hand	Reorder_lvl	Sell_price	Cost_price
------------	-------------	-----------	-----------	-------------	-------------	------------	------------

15) Create table cmaster from client\_master table.

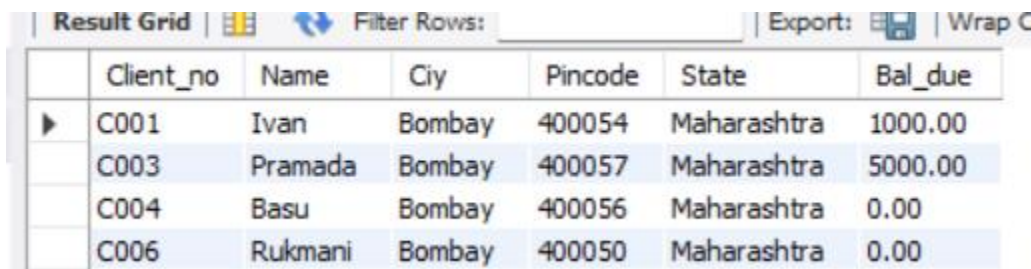
```
create table cmaster like client_master;
```

```
truncate cmaster;
```

```
drop table cmaster;
```

```
create table cmaster select * from client_master where City='bombay';
```

```
select * from cmaster;
```



The screenshot shows a 'Result Grid' with columns: Client\_no, Name, Ciy, Pincode, State, and Bal\_due. It contains four rows of data for clients in Bombay.

Client_no	Name	Ciy	Pincode	State	Bal_due
C001	Ivan	Bombay	400054	Maharashtra	1000.00
C003	Pramada	Bombay	400057	Maharashtra	5000.00
C004	Basu	Bombay	400056	Maharashtra	0.00
C006	Rukmani	Bombay	400050	Maharashtra	0.00

16) Insert data in cmaster from client\_master where city='bombay'

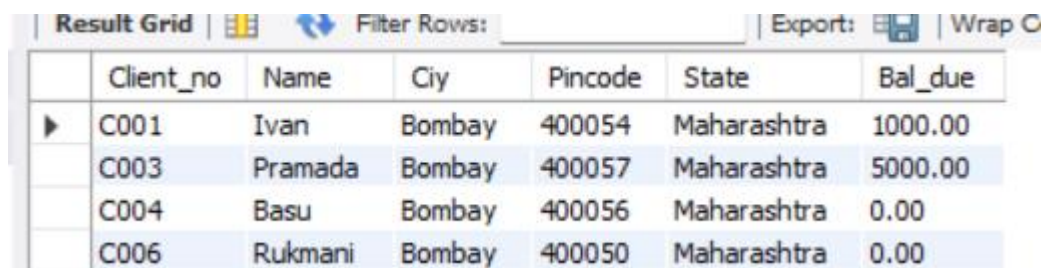
```
create table cmaster like client_master;
```

```
truncate cmaster;
```

```
drop table cmaster;
```

```
create table cmaster select * from client_master where City='bombay';
```

```
select * from cmaster;
```



The screenshot shows a 'Result Grid' with columns: Client\_no, Name, Ciy, Pincode, State, and Bal\_due. It contains four rows of data for clients in Bombay.

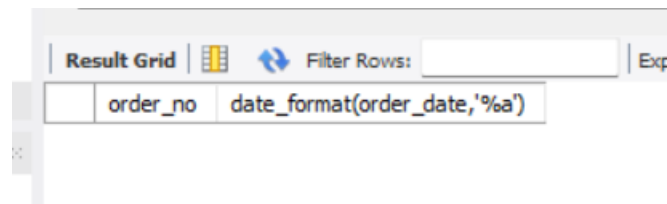
Client_no	Name	Ciy	Pincode	State	Bal_due
C001	Ivan	Bombay	400054	Maharashtra	1000.00
C003	Pramada	Bombay	400057	Maharashtra	5000.00
C004	Basu	Bombay	400056	Maharashtra	0.00
C006	Rukmani	Bombay	400050	Maharashtra	0.00

**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

17) Create table sales from sales\_order with order\_no and client\_no columns.

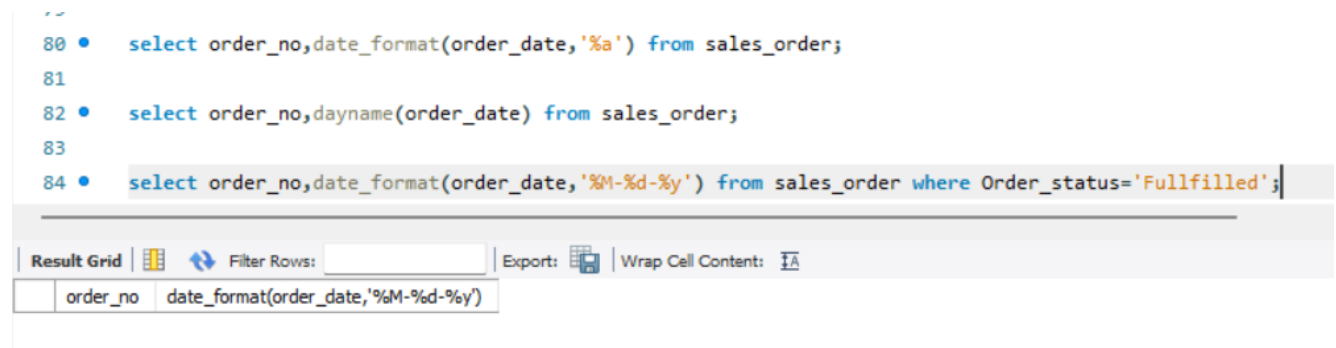
create table sales as select order\_no,client\_no from sales\_order;

select \* from sales;



order_no	date_format(order_date, '%a')
----------	-------------------------------

18) Insert data in sales from sales\_order table.



```
80 • select order_no,date_format(order_date,'%a') from sales_order;
81
82 • select order_no,dayname(order_date) from sales_order;
83
84 • select order_no,date_format(order_date,'%M-%d-%y') from sales_order where Order_status='Fullfilled';
```

order_no	date_format(order_date, '%M-%d-%y')
----------	-------------------------------------

### Queries on Date manipulation:

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

```
select order_no,date_format(order_date,'%a') from sales_order;
select order_no,dayname(order_date) from sales_order;
```

	order_no	date_format(order_date, '%a')
▶	O1901	Fri
	O1902	Sun
	O4665	Wed
	O1903	Fri
	O4666	Wed
	O1908	Wed

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



**Name: Ayush Patel    Class B batch 35    Enrollment : 22162171038**

3) Find the number of days elapsed between delivery date and order date from sales\_order table.

```
select datediff(order_date,dely_date) from sales_order;
```

	datediff(order_date,dely_date)
▶	-8
	-153
	-2
	-4
	-2
	18

4) Find the date, 15 days after today's date.

88

```
89 • select adddate(current_date(),15)from dual;
```

Result Grid	Filter Rows:	Export:	Wrap
	adddate(current_date(),15)		
▶	2023-08-10		

5) Display current date and time.

```
91 • select now() from dual;
```

Result Grid	Filter Rows:
	now()
▶	2023-07-26 20:20:26

6) Display system time.

74

```
93 • select sysdate() from dual;
```

Result Grid	Filter Rows:
	sysdate()
▶	2023-07-26 20:20:44