Unit-6: Retrieving Data using SQL	
Short Questions [1 Mark]	
1.	**List at least six character functions.**
Ans:	➢ The character functions are as follows: 1. ASCII() 2. CONCAT() 3. INITCAP() 4. INSTR() 5. LENGTH() 6. LOWER() 7. UPPER() 8. LPAD() 9. RPAD() 10. TRIM() 11. LTRIM() 12. RTRIM() 13. REPLACE()
2.	**List at least six numeric functions supported by MySQL.**
Ans:	➢ The numeric functions supported by MySQL are: 1. ABS() 2. POWER() 3. ROUND() 4. SQRT() 5. EXP() 6. GREATEST() 7. LEAST() 8. TRUNCATE() 9. FLOOR() 10. CEIL()
3.	**List the different aggregate functions.**
Ans:	➢ The different aggregate functions are: 1. Average: AVG() 2. Count: COUNT() 3. Maximum: MAX() 4. Minimum: MIN() 5. Total: SUM()
4.	**By using which function, we can determine the current user connected to MySQL from a particular host?**
Ans:	➢ USER() is used to determine the current user connected to MySQL from a particular host.

	➢ For Example: Select USER();
5.	**What is the use of "GROUP BY" clause?**
Ans:	➢ "Group by" clause is used to apply aggregate functions to a set of tuples. ➢ The attributes given in the group by clause are used to form groups. ➢ Tuples with the same value on all attributes in the group by clause are placed in one group.
6.	**What are Joins?**
Ans:	➢ An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.
7.	**List different types of Joins.**
Ans:	➢ Different types of Joins are as follows: **1.** Natural Join (inner join) **2.** Multiple Table Join (outer join) **3.** Cartesian Join **4.** Self Join
8.	**How Natural Joins are implemented?**
Ans:	The natural joins are implemented using three possible join clauses that use the following keywords in different combinations: **1.** NATURAL JOIN **2.** JOIN…..USING **3.** JOIN….ON
9.	**Which clause in SQL is used to implement JOIN operation in relational algebra?**
Ans:	"FROM" clause in SQL is used to implement JOIN operation in relational algebra.
10.	**What are set operators?**
Ans:	➢ Set operators are the operators which are used to combine the multiple queries into a single query. ➢ It is used to control the order of rows returned.
11.	**Which set operators will remove duplicate rows from the final result?**
Ans:	UNION, INTERSECT and MINUS operators will remove duplicate rows from the final result.
12.	**Which set operator will combine all the rows without eliminating duplication in the final result?**
Ans:	"Union All" operator will combine all the rows without eliminating duplication in the final result.

13. What is INTERSECT operator?

Ans:
- The INTERSECT operator is used to return rows that are common in both queries, sorting them and removing duplications.

14. What is MINUS operator?

Ans:
- The MINUS operator is used to return rows in the first query that are not present in the second query, sorting them and removing duplications.

15. When it is required to use self-join?

Ans: A self-join is required when the join columns originate from the same table

Short Questions [2 Marks]

1. What is the difference between INITCAP() and UPPER()?

Ans:

INITCAP()	UPPER()
1. It is use to return a string with the first letter of each word in upper case. 2. Syntax: INITCAP(char) 3. For Example: SELECT INITCAP('mscit') "Title Case";	1. It is use to return all the character letters in upper case. 2. Syntax: UPPER(char) 3. For Example: SELECT UPPER('mscit') "Upper";

2. Explain REPLACE function with syntax and example.

Ans:
- Replace takes three arguments,

Syntax:

REPLACE(string1, string_to_replace, [replacement_string])

- where string1, string_to_replace and replacement_string are character strings.

- This function returns string1 with all occurrence of , string_to_replace replaced with replacement_string. replacement_string defaults to NULL.
- If replacement_string is null, all occurrences of , string_to_replace are removed.
- If string_to_replace is null, then string1 is returned unchanged.
- If string1 is null, then NULL is returned.

- For Example:

```
SELECT  REPLACE('*SHAL','*','UI');
```

3. Define CEIL() and FLOOR() with example.

Ans:

1. CEIL():

- CEIL() is used to return the smallest integer value that is greater than or equal to a number.
- For Example:

SELECT CEIL(24.8) "Ceil1", CEIL(13.15) "Ceil2";

Ceil1	Ceil2
25	14

2. FLOOR():

- FLOOR() is used to return a largest integer value that is equal to less than a number.
- For Example:

SELECT FLOOR(24.8) "Flr1", FLOOR(13.15) "Flr2";

Flr1	Flr2
24	13

4. State the use of ORDER BY clause by giving suitable example.

Ans:

The ORDER BY clause is used to sort the column in ascending or descending order.

- Syntax:

SELECT <columns> FROM <tables> WHERE <predicates> ORDER BY <column> ASC/DESC;

- For Example:

SELECT NAME FROM STUDENT ORDER BY NAME;

5. How "Having" clause is an additional filter to the "where" clause?

Ans:

- The SQL HAVING clause allows us to restrict the data.
- HAVING allows a user to perform conditional tests on aggregate values.
- It is often used in combination with GROUP BY clause.
- For example:

To find the employees who received more than Rs. 1,000 in bonuses for the year of 2016. So, you might think that we could write a query like:

```
select employee, sum(bonus) from emp_bonus
group by employee where sum(bonus) > 1000;
```

- But Group functions or Aggregates function cannot be used in the WHERE clause, they are used only inside HAVING.
- Therefore, the correct query will be….

```
select employee, sum(bonus) from emp_bonus
group by employee having sum(bonus) > 1000;
```

- Therefore, we can say that "Having" clause is an additional filter to the "where" clause.

6. What are JOINS? Which are the different types of Joins?

Ans:

- SQL joins are used to combine rows from two or more tables.
- Different types of Joins are as follows:
  1. Natural Join
  2. Multiple Table Join
  3. Cartesian Join
  4. Self Join

7. What is the use of JOIN…USING clause?

Ans:

- The JOIN…USING clause allows one or more equijoin columns to be explicitly specified in brackets after the USING keyword.
- Syntax:

```
SELECT table1.column, table2.column FROM table1
JOIN table2 USING (join_column1, join_column2…);
```

- For Example:

```
select * from department join employee using(dept_id);
```

```
SQL> select * from department join employee using(dept_id);

   DEPT_ID DEPT_NAME           EMP_ID EMP_NAME
---------- --------------- ---------- ----------
     20111 purchase                 1 raj
     20114 finance                  3 shivang
     20114 finance                  4 darshan
     20113 accounting               5 sapna
     20113 accounting               6 yash
     20116 manufacture              8 ron

6 rows selected.
```

8. How can we join a table using the JOIN…ON clause?

Ans:

- The JOIN…ON clause allows the explicit specification of join columns, regardless of their column names.

- This is the most flexible and widely used form of the join clauses.
- The ON and NATURAL keywords cannot appear together in a join clause.

- Syntax:

  SELECT table1.column, table2.column FROM

  table1 JOIN table2 ON (table1.column_name = table2.column_name);

- For Example:

  select * from department d join employee e on (d.dept_id = e.dept_id);

```
SQL> select * from department d join employee e
  2     on (d.dept_id = e.dept_id);

   DEPT_ID DEPT_NAME           EMP_ID EMP_NAME      DEPT_ID
---------- -------------- ---------- ---------- ----------
     20111 purchase                1 raj            20111
     20114 finance                 3 shivang        20114
     20114 finance                 4 darshan        20114
     20113 accounting              5 sapna          20113
     20113 accounting              6 yash           20113
     20116 manufacture             8 ron            20116

6 rows selected.
```

9. What is the difference between LEFT OUTER JOIN and RIGHT OUTER JOIN?

Ans:

LEFT OUTER JOIN	**RIGHT OUTER JOIN**
➢ A left outer join between the source and target tables returns the results of an inner join and the missing rows it excluded from the source table.	➢ A right outer join between the source and target tables returns the results of an inner join and the missing rows it excluded from the target table.

10. How can we join two tables by using FULL OUTER JOINS?

Ans:

- A FULL OUTER JOIN returns the combined results of a LEFT OUTER JOIN and RIGHT OUTER JOIN.
- The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).

- For Example:

```
select d.dept_id, d.dept_name, e.emp_id, e.emp_name
from department d full outer join employee e
on (d.dept_id = e.dept_id);
```

11.	**What is Cartesian join? Give one example.** **or** **Define Cross join with example.**
Ans:	➢ Cartesian join is also known as Cross Join. ➢ A Cartesian product of two tables may be conceptualized as joining each row of the 1st table with every row in the 2nd table. ➢ Syntax: SELECT table1.column, table2.column FROM table1 CROSS JOIN table2; ➢ For Example: select * from department d cross join employee e where d.dept_id='20114'; SQL> select * from department d cross join employee e 2 where d.dept_id='20114'; DEPT_ID DEPT_NAME EMP_ID EMP_NAME DEPT_ID 20114 finance 1 raj 20111 20114 finance 2 rohit 20114 finance 3 shivang 20114 20114 finance 4 darshan 20114 20114 finance 5 sapna 20113 20114 finance 6 yash 20113 20114 finance 7 mohit 20114 finance 8 ron 20116 8 rows selected.
12.	**What are set operators? List all four set operators.**
Ans:	➢ Set operators are the operators which are used to combine the multiple queries into a single query. ➢ The four set operators are: 1. UNION 2. UNION ALL 3. INTERSECT 4. MINUS
13.	**What is the use of Union and intersection operation?**
Ans:	➢ The UNION operator returns results from both queries, sorting them and after that it eliminates the duplications. ➢ The INTERSECT operator returns rows that are common in both queries, sorting them and removing duplications.

14. **What is the difference between UNION and UNION ALL?**

Ans:

UNION	UNION ALL
1. The UNION operator returns results from both queries, sorting them and after that it eliminates the duplications.	1. The UNION ALL operator returns results from both queries without sorting or removing duplications.
2. For Example: SELECT employee_id, job_id FROM employees UNION SELECT employee_id, job_id FROM job_history;	2. For Example: SELECT employee_id, job_id, department_id FROM employees UNION ALL SELECT employee_id, job_id, department_id FROM job_history;

15. **When it is required to use self-join?**

Ans:

- A self-join is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
- This can be useful when modeling hierarchies.
- They are also useful for comparisons within a table.

- For example:

```
SELECT  a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

Scenario based Questions [5 Marks]

1. **Write the MySQL queries for the following:**

a. **Find the length of the string 'BCA Honors, Bardoli'.**
b. **Display the string 'BCA Honors, Bardoli' in lower case.**
c. **Display the string 'BCA Honors, Bardoli' in upper case.**
d. **Display the string 'BCA Honors, Bardoli' with the first letter of each word in upper case.**
e. **Display only the string 'Honors' from the string 'BCA Honors, Bardoli'.**

Ans:

a. select length('BCA Honors, Bardoli');

```
LENGTH('BCAHONORS,BARDOLI')
--------------------------
                        19
```

b. select lower('BCA Honors, Bardoli');

```
LOWER('BCAHONORS,BA
-------------------
bca honors, bardoli
```

c. select upper('BCA Honors, Bardoli');

```
UPPER('BCAHONORS,BA
-------------------
BCA HONORS, BARDOLI
```

d. select initcap('BCA Honors, Bardoli');

```
INITCAP('BCAHONORS,
-------------------
Bca Honors, Bardoli
```

e. select substr('BCA Honors, Bardoli', 5, 6);

```
SUBSTR(
-------
Honors
```

2. Write the MySQL queries for the following:

a. **To display the date of the day after three months of today's date.**
b. **Find the difference of the months between MAR-2016 and SEP-2016.**
c. **Display today's date.**
d. **To display the last day of current month.**
e. **To display the next date from today's date.**

Ans: **a.**

```
mysql> select DATE_ADD(NOW(), interval 3 month);
+-----------------------------------+
| DATE_ADD(NOW(), interval 3 month) |
+-----------------------------------+
| 2017-02-15 09:06:18               |
+-----------------------------------+
1 row in set (0.00 sec)
```

b.

```
mysql> select PERIOD_DIFF('201609','201603') 'Months';
+--------+
| Months |
+--------+
|      6 |
+--------+
1 row in set (0.00 sec)
```

c.

```
mysql> select NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2016-11-15 09:20:37 |
+---------------------+
1 row in set (0.00 sec)
```

OR

```
mysql> select SYSDATE();
+---------------------+
| SYSDATE()           |
+---------------------+
| 2016-11-15 09:20:55 |
+---------------------+
1 row in set (0.00 sec)
```

d.

```
mysql> select LAST_DAY(NOW()) 'Last Day';
+------------+
| Last Day   |
+------------+
| 2016-11-30 |
+------------+
1 row in set (0.00 sec)
```

e.

```
mysql> select ADDDATE(CURRENT_DATE(),1);
+---------------------------+
| ADDDATE(CURRENT_DATE(),1) |
+---------------------------+
| 2016-11-16                |
+---------------------------+
1 row in set (0.02 sec)
```

3. **Consider the following table and write the MySQL queries:**

tblBankAccount(ano, balance, branch)

a. **Find maximum balance of the bank account.**
b. **Find minimum balance of the bank account.**
c. **Find the total balance of the bank.**
d. **Find the average balance of the bank.**
e. **Find the total number of accounts in a bank.**

Ans:

a. **select max(balance) "Maximum" from tblBankAccount;**

```
Maximum
----------
845000
```

b. **select min(balance) "Minimum" from tblBankAccount;**

```
Minimum
----------
2000
```

c. **select sum(balance) "Total" from tblBankAccount;**

```
Total
----------
27880
```

d. **select avg(balance) "Average" from tblBankAccount;**

```
Average
----------
6970
```

e. **select count(balance) "Total Accounts" from tblBankAccount;**

```
Total Accounts
--------------
4
```

4. Consider the following table and write the MySQL queries:

tblEmployee(eid, fname, lname, address, basic, bonus)

a. **Show the maximum Basic salary of the employees, whose Bonus is greater than 40.**
b. **Display the first name and the last name of employees using Title Case.**
c. **To count the number of employees, whose department is "Sales".**

d. Remove the space from both the sides of the employee address and display whole table.

e. Calculate the total salary from basic and bonus and find the average of it.

Ans:

a. Select max(basic) from tblEmployee where bonus>40;

b. Select initcap(fname), initcap(lname) from tblEmployee;

c. Select count(*) from tblEmployee where department=”Sales”;

d. Select eid, fname, lname, trim(address), basic, bonus from tblEmployee;

e. Select avg(basic+bonus) from tblEmployee;

5. Consider the following tables STOCK and Dealers and give the output of the following queries:

Table: STOCK

Itemno	Item	dcode	Qty	unitprice	stockdate
5005	Ball Pen 0.5	102	100	16	31-Mar-10
5003	Ball Pen 0.25	102	150	20	01-Jan-10
5002	Gel Pen Premium	101	125	14	14-Feb-10
5006	Gel Pen Classic	101	200	22	01-Jan-10
5001	Eraser Small	102	210	5	19-Mar-09
5004	Eraser Big	102	60	10	12-Dec-09
5009	Sharper Classic	103	160	8	23-Jan-09

Table: DEALERS

Dcode	Dname
101	Reliable Stationers
103	Classic Plastics
102	Clear Deals

a. Select count(distinct dcode) from stock;

b. Select qty*unitprice from stock where itemno = 5006;

c. Select item, dname from stock s, dealers d where s.dcode = d.dcode and itemno = 5004;

d. Select min(stockdate) from stock;

e. Select item from stock where item like ‘s%’.

Ans:

a. 3

b. 4400

c. Eraser Big

Clear Deals

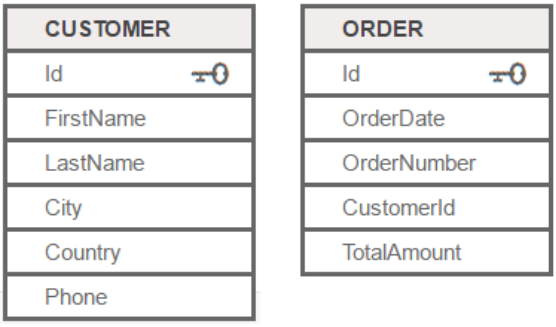
d. 23-Jan-09

e.

Item

Sharper Classic

6. Consider the following tables and write the queries for the following:



a. List total customers in each country. Display results with easy to understand column headers.

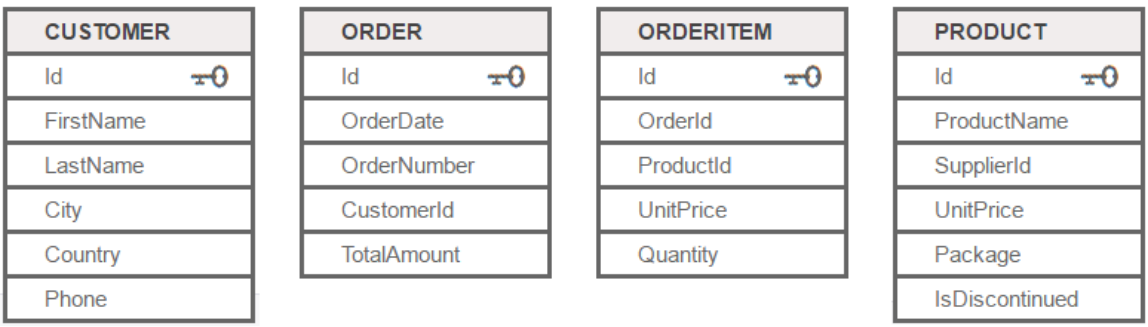
b. List customers that have not placed orders.

Ans:

a.
```
SELECT COUNT(C.Id) AS TotalCustomers, C.Country AS Nation
FROM Customer C
GROUP BY C.Country;
```

b.
```
SELECT TotalAmount, FirstName, LastName, City, Country
FROM Order O RIGHT JOIN Customer C
ON O.CustomerId = C.Id
WHERE TotalAmount IS NULL;
```

7. Consider the following tables and write the queries for the following:



a. List all orders with customer information.

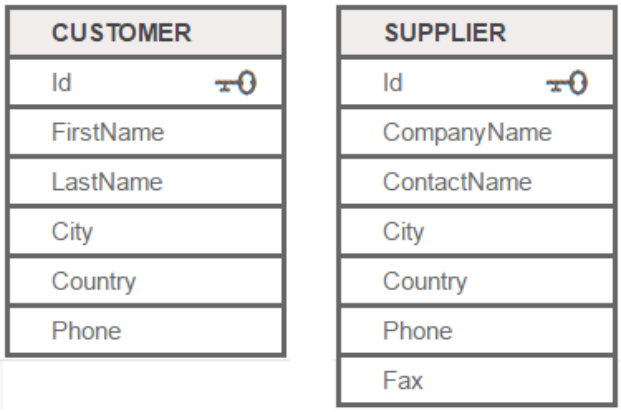
b. List all orders with product names, quantities and prices.

Ans:

a.
```
SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
FROM Order O JOIN Customer
ON O.CustomerId = Customer.Id;
```

b.
```
SELECT O.OrderNumber, CONVERT(date,O.OrderDate) AS Date,
P.ProductName, I.Quantity, I.UnitPrice
FROM Order O JOIN OrderItem I
ON O.Id = I.OrderId
JOIN Product P
ON P.Id = I.ProductId
ORDER BY O.OrderNumber;
```

8. Consider the following tables and write the queries for the following:



a. To match all the customers and suppliers by country.

b. List all contacts of suppliers and customers.

Ans:

a.
```
SELECT C.FirstName, C.LastName, C.Country AS CustomerCountry,
S.Country AS SupplierCountry, S.CompanyName
FROM Customer C FULL JOIN Supplier S
ON C.Country = S.Country;
```

b.
```
SELECT 'Customer' As Type, Phone FROM Customer
UNION
SELECT 'Supplier', Phone FROM Supplier;
```

Long Questions [5 Marks]	
1.	**Which are the two different types of SQL FUNCTION? Explain any one by giving example.**

Ans:

- There are two type of SQL functions:
  1. Single row function
  2. Multi row function

1. **Single row functions:**

1. LOWER:
   - It is use to return all the character letters or a string in lower case.
   - Syntax:

     LOWER(char/string)

   - For Example:

```
SQL> select lower(fname), lname, salary from emp;

LOWER(FNAME)         LNAME                SALARY
-------------------- -------------------- ----------
anjali               Rathod                10000
afzal                Vadi                  15000
preeti               Sangtani              20000
foram                Sheth                 50000
rizwan               Patel                 18000
priyanka             Ahir                  25000
```

2. UPPER:
   - It is use to return all the character letters or string in upper case.
   - Syntax:

     UPPER(char/string)

   - For Example:

```
SQL> select upper(fname), lname, salary from emp;

UPPER(FNAME)         LNAME                SALARY
-------------------- -------------------- ----------
ANJALI               Rathod                10000
AFZAL                Vadi                  15000
PREETI               Sangtani              20000
FORAM                Sheth                 50000
RIZWAN               Patel                 18000
PRIYANKA             Ahir                  25000
```

3. INITCAP:
   - It is use to return a string with the first letter of each word in upper case.
   - Syntax:

     INITCAP(char/string)

- For Example:

```
SQL> select initcap(fname), initcap(lname), salary from emp;

INITCAP(FNAME)       INITCAP(LNAME)           SALARY
-------------------- -------------------- ----------
Anjali               Rathod                    10000
Afzal                Vadi                      15000
Preeti               Sangtani                  20000
Foram                Sheth                     50000
Rizwan               Patel                     18000
Priyanka             Ahir                      25000
```

4. CONCAT:
   - The CONCAT function joins two character literals, columns, or expressions to yield one larger character expression.
   - Numeric and date literals are implicitly cast as characters, when they occur as parameters to the CONCAT function.
   - Syntax:

     CONCAT(string1, string2)

   - For Example:

```
mysql> select concat(eno,name) from student;
+------------------+
| concat(eno,name) |
+------------------+
| 1Jigar           |
| 3Dinu            |
| 4Jil             |
| 5Bhavik          |
+------------------+
```

5. LENGTH():
   - It is use to return the length of a word.
   - Syntax:

     LENGTH(word)

   - For Example:

```
mysql> select length('bca honors');
+----------------------+
| length('bca honors') |
+----------------------+
|                   10 |
+----------------------+
1 row in set (0.00 sec)
```

2. **Explain the following character functions with its usages, syntax and example:**
LPAD(), RPAD(), TRIM()

Ans: **1. LPAD:**

- It is use to return "char1", left-padded to length "n" with the sequence of characters specified in "char2".

- If “char2” is not specified oracle uses blanks by default.

- Syntax:

  LPAD (char1, n , [char2]);

  - Where, “char1” and “char2” are character string and “n” is an integer.

- For Example:

  SELECT LPAD(last_name,10) ‘lpad_lname’, LPAD(salary,8,'*') ‘lpad_salary’ FROM employees WHERE last_name like 'J%';

LPAD_LNAME	LPAD_SAL
Johnson	****6200
Jones	****2800

2. RPAD:

- It is use to return “char1”, right-padded to length “n” with the characters specified in “char2”.
- If “char2” is not specified oracle uses blanks by default.

- Syntax:

  LPAD (char1, n , [char2]);

  - Where, “char1” and “char2” are character string and “n” is an integer.
  - If “char1” is more than padded_length character, it is truncated to padded_length character. pad_string defaults to a single space string1.

- For Example:

SELECT RPAD(last_name,10) ‘rpad_rname’, RPAD(salary,8,'*') ‘rpad_salary’ FROM employees WHERE last_name like 'J%';

RPAD_RNAME	RPAD_SAL
Johnson	6200****
Jones	2800****

3. TRIM:

- It is use to remove all specified characters either form the beginning of the ending of a string.
- Syntax:

  TRIM([leading | trailing | both [<trim_character> from]]<string1>)

  - *leading* – remove trim_string from the front of string1.
  - *trailing* – remove trim_string from the end of the srting1.
  - *Both* – remove trim_string from the front and end of string1.

- If none of the above option is chosen, the TRIM function will remove trim_string from both the front and end of sting1.
  - *trim_character* is the character that will be removed from string1.
  - If this parameter is omitted, the trim function will remove all *leading* and *trailing* spaces from string1.
  - string1 is the string to trim.

- For Example:

(1) SELECT trim(leading '1' from '1VISHAL');

```
TRIM(T
---------
VISHAL
```

(2) SELECT trim(trailing '1' from 'VISHAL1');

```
TRIM(T
---------
VISHAL
```

(3) SELECT trim(both '1' FROM '1VISHAL');

```
TRIM(B
---------
VISHAL
```

3. Describe the following character functions with its use, syntax, and example.

INITCAP(), LTRIM(), RTRIM(), REPLACE()

Ans:

1. **INITCAP():**
   - It is use to return a string with the first letter of each word in upper case.
   - Syntax:

     INITCAP(char)
   - For Example:

     SELECT INITCAP('bca honors') "Title Case";

2. **LTRIM():**
   - It is use to return the string *str* with leading space characters removed.
   - Syntax:

     LTRIM(str)
   - For Example:

     SELECT LTRIM(' TARSADIA') 'Left Trim of Tarsadia';

3. **RTRIM():**
   - It is use to return the string *str* with trailing space characters removed.
   - Syntax:

     RTRIM(str)
   - For Example:

     SELECT RTRIM('TARSADIA ') "Right Trim of Tarsadia";

4. **REPLACE():**
   - The REPLACE function replaces all occurrences of a search item in a source string with a replacement term and returns the modified source string.
   - If the length of the replacement term is different from that of the search item, then the lengths of the returned and source strings will be different.
   - If the search string is not found, the source string is returned unchanged.
   - The REPLACE function takes three parameters.
   - If the replacement term parameter is omitted, each occurrence of the search item is removed from the source string.
   - In other words, the search item is replaced by an empty string.

   - Syntax:

     REPLACE(source string, search item, <replacement term>);

- For Example:

```
SELECT  REPLACE( ' * SHAL ' , ' * ' , ' VI ');
SELECT REPLACE(SYSDATE, 'DEC','NOV');
SELECT REPLACE('1#3#6#7#9#','#','->');
```

4. List and example any five date functions with proper syntax and example.

Ans: The date functions are as follows:

1. NOW()
2. CURDATE()
3. CURTIME()
4. DATE_ADD / DATE_SUB
5. ADDDATE / SUBDATE
6. LAST_DAY()
7. PERIOD_DIFF()

1. NOW():

- Now() is a function that requires no arguments and returns the current date and time.
- Syntax:

```
NOW()
```

- For Example:

```
SELECT NOW();
```

2. CURDATE():

- It returns the current date as a value in YYYY-MM-DD or YYYYMMDD format.
- The format depends on whether the function is used with a date expressed as a string or numeric value.
- It takes no arguments.
- For example:

```
SELECT CURRENT_DATE();
SELECT CURRENT_DATE() + 0;
```

3. DATE_ADD / DATE_SUB():

- It is used to return a date after adding / subtracting the value specified as a parameter to the function.

- Syntax:

DATE_ADD(<date>, INTERVAL <value> <type>)

DATE_SUB(<date>, INTERVAL <value> <type>)

- For Example:

SELECT CURRENT_DATE, DATE_ADD(CURRENT_DATE, INTERVAL 2 MONTH);

4. LAST_DAY():

- It is used to return the last date of the month passed as a parameter to the function.
- Syntax:

LAST_DAY(<Date>)

- For Example:

SELECT CURRENT_DATE, LAST_DAY(CURRENT_DATE) "Last Date";

5. PERIOD_DIFF():

- It is used to return the number in months between two given date values.
- Syntax:

PERIOD_DIFF(<Period1>,<Period2>)

  - Where, Period1 and Period2 should be either in YYYYMM or YYMM format only.

- For Example:

SELECT PERIOD_DIFF('200507','200401') "Months";

5. Explain the use of "Select" and "Where" clause with their syntax and example.

Ans:

SELECT:

- SELECT allows you to retrieve information already stored in the database.
- The statement begins with keyword SELECT, followed by the column names whose data you want to query.

- Syntax:

SELECT column_name(s)FROM table_name

OR

SELECT * FROM TABLE_NAME;

- For Example:

```
SELECT NAME FROM STUDENT ;

NAME
----------
Jigar
Shivani
Denish
Jil
Piyar
```

- **WHERE:**
  - The where clause is used in the select statement to limit the number of rows processed.
  - Any logical condition of the where clause use the comparisons operator.
  - Rows are returned or operated upon where the data satisfied the logical conditions(s) of the where clause.
  - You can use column names or expressions in the where clause, but not column alias names.
  - The where clause follows the form clause in the select statement.

- Syntax:

```
SELECT column_name(s) FROM table_name WHERE column_name operator value
```

- For Example:

```
SELECT NAME FROM STUDENT WHERE CITY = ' NAVSARI ' ;

NAME
----------
Shivani
Rizwan
Priti
Vishal
```

6. What are the difference between Aggregate function and Scalar function?

or

Differentiate Group function and Single Row function.

Ans:

Aggregate function (Group function)	Scalar function (Single Row function)
1. The Functions that operate on a set of rows (or values) are called Aggregate or Group functions.	1. The Functions that operate on a single row (or value) are called Scalar or Single Row functions.
2. These functions accept a set of rows, i.e. a group, as an input and return a single row as a result.	2. These functions return one result for every row given as an input.
3. If five rows are given as an input, there will be only single result as an output.	3. If five rows are given as an input, there will be five results as an output.
4. For Example: MAX() is an aggregate function. It finds maximum number out of set of numbers, and returns single value as an answer.	4. For Example: Length is a scalar function. It finds length of a given string and returns individual results for each row given as an input.
5. Useful aggregate functions: ▪ AVG() - Returns the average value. ▪ COUNT() - Returns the number of rows. ▪ MAX() - Returns the largest value. ▪ MIN() - Returns the smallest value. ▪ SUM() - Returns the sum.	5. Useful scalar functions: ▪ UPPER() - Converts a field to upper case. ▪ LOWER() - Converts a field to lower case. ▪ LENGTH() - Returns the length of a text field.

7. List and explain all aggregate functions with example.

Ans:

- The aggregate function are as follows:
  1. COUNT()
  2. SUM()
  3. AVG()
  4. MAX()
  5. MIN()

1. COUNT:

- It count the number of values in the particular column which is write in the "()" braces.
- For Example:

```
SELECT COUNT (SID) FROM SAILORS;
```

2. **SUM:**
   - It finds the sum of the all values of the particular column.
   - For Example:

SELECT SUM (RATING) FROM SAILORS;

3. **AVG:**
   - It finds the average of the all values of the particular column.
   - For Example:

SELECT AVG (AGE) FROM SAILORS;

4. **MAX:**
   - It finds the maximum value of the particular column.
   - For Example:

SELECT MAX (AGE) FROM SAILORS;

5. **Min:**
   - It finds the minimum value of the particular column.
   - For Example:

SELECT MIN (AGE) FROM SAILORS;

8. Explain Group By and Having clause with their use, syntax, options/operators and example.

Ans:

GROUP BY:

- A The Group By clause is used in SELECT statements to divide the table into groups.
- Grouping can be done by a column name or with aggregate functions in which case the aggregate produces a value for each group.
- If no GROUP BY clause is specified, the default groupings become the entire result set.
- When the query executed and the data is fetched, it is grouped based on the group by clause, and the group function is applied.

- Syntax:

SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name

- We normally use a GROUP By clause in conjunction with an aggregate expression (like SUM, COUNT, etc.)
- Options:
  - SUM
  - COUNT
  - MIN
  - MAX
  - AVG

- For Example:

```
SELECT NAME, AVG( RESULT ) FROM STUDENT WHERE
CITY = ’  NAVSARI’
GROUP BY NAME;
```

NAME	SUM (RESULT)
JIGAR	75

HAVING:

- The SQL HAVING clause allows us to restrict the data that is sent to the GROUP BY clause.
- Group functions cannot be used in the WHERE clause. SQL statement can have both a WHERE clause and an HAVING clause. WHERE filters data before grouping and after grouping.
- A select statement includes a HAVING clause to filter the grouped data.
- HAVING allows a user to perform conditional tests on aggregate values. It is often used in combination with GROUP BY.
- With HAVING, you can include or exclude groups based on the aggregate value for that group.

- Syntax:

```
SELECT column_name , aggregate_function ( column_name ) FROM     table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function ( column_name )  operator value;
```

- Operator:
  - SUM
  - AVG
  - MIN
  - MAX
  - COUNT

- For Example:

```
SELECT  RESULT, SUM(RESULT) FROM STUDENT
WHERE CITY='NAVSARI'
GROUP BY RESULT
HAVING RESULT > 50;
```

RESULT	SUM(RESULT)
65	130

9. Which Join clauses are used in Natural Join? Explain each with examples.

Ans:

- The NATURAL JOIN is implemented using three possible join clauses that use the following keywords in different combinations:
  1. NATURAL JOIN
  2. USING
  3. ON

1. Natural Join using "Natural Join" Clause:

- The NATURAL JOIN identifies columns with common names between the two tables.
- Syntax:

```
SELECT table1.column, table2.column
FROM table1
NATURAL JOIN table2;
```

- For Example:
  - select * from locations natural join countries;

2. Natural Join using "JOIN…USING" Clause:

- The JOIN…USING clause allows one or more equijoin columns to be explicitly specified in brackets after the USING keyword.
- Syntax:

```
SELECT table1.column, table2.column
FROM table1
JOIN table2 USING (join_column1, join_column2…);
```

- For Example:
  - select * from locations join countries using (country_id);

3. Natural Join using "JOIN…ON" Clause:

- The JOIN…ON clause allows the explicit specification of join columns, regardless of their column names.
- This is the most flexible and widely used form of the join clauses.
- The ON and NATURAL keywords cannot appear together in a join clause.
- Syntax:

```
SELECT table1.column, table2.column
FROM table1
JOIN table2 ON (table1.column_name = table2.column_name);
```

- For Example:
  - select * from departments d join employees e
    on e.employee_id=d.department_id;

10. What is Outer Joins? Explain different types of outer joins by giving examples.

Ans: Outer Joins:

- The multiple table join is also known as outer join.
- Often in joining two tables we find that a row in one table that does not have a matching row in the other table.
- For example, a several CUSTOMER_ID numbers may not appear in the ORDER table.
- As a result, the EQUI-JOIN and NATURAL JOIN do not include all of the customers in CUSTOMER.

- Using an OUTER JOIN rows that do not have matching values in common columns are also included in the result table.
- Null values appear in columns, where there is not a match between the tables.
- Outer Loins are of different types:
    1. LEFT OUTER JOIN
    2. RIGHT OUTER JOIN
    3. FULL OUTER JOINS

1. Left Outer Join:

- The LEFT OUTER JOIN keyword is used to return all the rows from the left table (table1), with the matching rows in the right table (table2).
- The result is NULL in the right side, when there is no match.
- A left outer join performs an inner join of table1 and table2 based on the condition specified after the ON keyword.
- Syntax:

```
SELECT table1.column, table2.column
FROM table1
LEFT OUTER JOIN table2
        ON (table1.column = table2.column);
```

✓ Consider the following tables:

Table 1: Department

dept_id	dept_name
20111	Purchase
20112	Sales
20113	Accounting
20114	Finance
20115	Marketing
20116	Manufacture

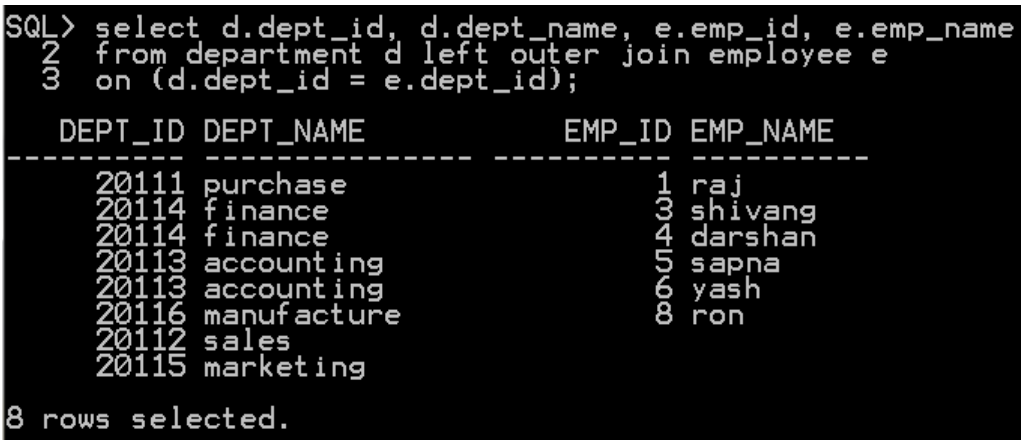
Table 2: Employee

emp_id	emp_name	dept_id
001	Raj	20111
002	Rohit	
003	Shivang	20114
004	Darshan	20114
005	Sapna	20113
006	Yash	20113
007	Mohit	
008	Ron	20116

- For Example:
  - List the department details and employee details for all departments. Include employee information even for departments that do have employee.
- Query:

```
select d.dept_id, d.dept_name, e.emp_id, e.emp_name
from department d left outer join employee e
on (d.dept_id = e.dept_id);
```

- Output:



```
SQL> select d.dept_id, d.dept_name, e.emp_id, e.emp_name
  2  from department d left outer join employee e
  3  on (d.dept_id = e.dept_id);

   DEPT_ID DEPT_NAME           EMP_ID EMP_NAME
---------- -------------- ---------- ----------
     20111 purchase                1 raj
     20114 finance                 3 shivang
     20114 finance                 4 darshan
     20113 accounting              5 sapna
     20113 accounting              6 yash
     20116 manufacture             8 ron
     20112 sales
     20115 marketing

8 rows selected.
```

- Note:
  - The LEFT OUTER JOIN keyword returns all the rows from the left table (department), even if there are no matches in the right table (employee).

2. Right Outer Joins:

- The RIGHT OUTER JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1).
- The result is NULL in the left side when there is no match.
- A right outer join performs an inner join of table1 and table2 based on the join condition specified after the ON keyword.
- Syntax:

```
SELECT table1.column, table2.column
FROM table1 RIGHT OUTER JOIN table2
ON (table1.column = table2.column);
```

- For Example:
  - List the department details and employee details for all department. Include department information even for employees that do have department using Right Outer Join.

- Query:

```
select d.dept_id, d.dept_name, e.emp_id, e.emp_name
from department d right outer join employee e
on (d.dept_id = e.dept_id);
```

- Output:

```
SQL> select d.dept_id, d.dept_name, e.emp_id, e.emp_name
  2  from department d right outer join employee e
  3  on (d.dept_id = e.dept_id);

   DEPT_ID DEPT_NAME           EMP_ID EMP_NAME
---------- --------------- ---------- ----------
     20111 purchase                 1 raj
     20113 accounting               6 yash
     20113 accounting               5 sapna
     20114 finance                  4 darshan
     20114 finance                  3 shivang
     20116 manufacture              8 ron
                                    7 mohit
                                    2 rohit

8 rows selected.
```

- Note:
  - The RIGHT JOIN keyword returns all the rows from the right table (employee), even if there are no matches in the left table (department).

3. Full Outer Joins:

- A FULL OUTER JOIN returns the combined results of a LEFT OUTER JOIN and RIGHT OUTER JOIN.
- The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).

- Syntax:

```
SELECT table1.column, table2.column
FROM table1 FULL OUTER JOIN table2
ON (table1.column = table2.column);
```

- For Example:
  - List the employee details and department details for all departments. Include department information even for employees that do have department using Full Outer Join.

- Query:

```
select d.dept_id, d.dept_name, e.emp_id, e.emp_name
from department d full outer join employee e
on (d.dept_id = e.dept_id);
```

- Output:

```
SQL> select d.dept_id, d.dept_name, e.emp_id, e.emp_name
  2  from department d full outer join employee e
  3  on (d.dept_id = e.dept_id);

   DEPT_ID DEPT_NAME           EMP_ID EMP_NAME
---------- --------------- ---------- ----------
     20111 purchase                 1 raj
     20114 finance                  3 shivang
     20114 finance                  4 darshan
     20113 accounting               5 sapna
     20113 accounting               6 yash
     20116 manufacture              8 ron
     20112 sales
     20115 marketing
                                    2 rohit
                                    7 mohit
```

- Note:
  - The FULL OUTER JOIN keyword returns all the rows from the left table (department), and all the rows from the right table (employee).
  - If there are rows in "department" that do not have matches in "employee", or if there are rows in "employee" that do not have matches in "department", those rows will also be listed.

11. Which are the different set operators in SQL? Demonstrate how they are used in query giving examples.

Ans:

- The set operators used in compound queries are as follows:
  1. UNION
  2. UNION ALL
  3. INTERSECT
  4. MINUS

- Construct queries involving the union, intersect, and minus operations over two sets by the following relation:

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

1. **Union Operation:**

- Returns the combined rows from two queries, sorting them and removing duplicates.
- For example:
  - To find the set of all courses taught either in Fall 2009 or in Spring 2010, or both.
  - The query will be:

```
(select course_id from section
where semester = 'Fall' and year= 2009)
union
(select course_id from section
where semester = 'Spring' and year= 2010);
```

- The union operation automatically eliminates duplicates, unlike the select clause.
- Thus, using the section relation where two sections of CS-319 are offered in Spring 2010, and a section of CS-101 is offered in the Fall 2009 as well as in the Fall 2010 semester, CS-101 and CS-319 appear only once in the result, shown as follows:

course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

The result relation for *c1* union *c2*.

2. **Union All Operator:**

- Returns the combined rows from two queries without sorting or removing duplicates.
- If we want to retain all duplicates, we must write union all in place of union.
- For example:

```
(select course_id from section
where semester = 'Fall' and year= 2009)
union all
(select course_id from section
where semester = 'Spring' and year= 2010);
```

- The number of duplicate tuples in the result is equal to the total number of duplicates that appear in both c1 and c2. So, in the above query, each of CS-319 and CS-101 would be listed twice.

3. **Intersection Operator:**

- Returns only the rows that occur in both queries' result sets, sorting them and removing duplicates.
- For Example:
  - To find the set of all courses taught in the Fall 2009 as well as in Spring 2010.
  - The query will be:

```
(select course_id from section
where semester = 'Fall' and year= 2009)
intersect
(select course_id from section
where semester = 'Spring' and year= 2010);
```

RESULT:

course_id
CS-101

The result relation for *c1* intersect *c2*.

- The intersect operation automatically eliminates duplicates. For example, if it were the case that 4 sections of ECE-101 were taught in the Fall 2009 semester and 2 sections of ECE-101 were taught in the Spring 2010 semester, then there would be only 1 tuple with ECE-101 in the result.

4. **Minus Operator:**

- Returns only the rows in the first result set that do not appear in the second result set, sorting them and removing duplicates.
- For Example:
  - To find all courses taught in the Fall 2009 semester but not in the Spring 2010 semester.
  - The query will be:

```
(select course_id from section
where semester = 'Fall' and year= 2009)
minus
(select course_id from section
where semester = 'Spring' and year= 2010);
```

RESULT:

course_id
CS-347
PHY-101

- Except operation 7 outputs, all tuples from its first input that do not occur in the second input; that is, it performs set difference. The operation automatically eliminates duplicates in the inputs before performing set difference.
- For example, if 4 sections of ECE-101 were taught in the Fall 2009 semester and 2 sections of ECE-101 were taught in the Spring 2010 semester, the result of the except operation would not have any copy of ECE-101.