**Aim: - Implement Simple SQL Queries like single table retrieval**

1) Find out the names of all clients.

mysql> select name from Client\_master;

+-----------------+

| name |

+-----------------+

| Ivan Bayross |

| Vandana Saitwal |

| Pramada Jaguste |

| Basu Navindgi |

| Ravi Sreedharan |

| Rukmini |

+-----------------+

2) print the entire client\_master table.

mysql> select \* from Client\_master;

+-----------+-----------------+----------+----------+--------+-------------+---------+----------+

| Client\_no | Name | Address1 | Address2 | City | State | Pincode | Bal\_due |

+-----------+-----------------+----------+----------+--------+-------------+---------+----------+

| C00001 | Ivan Bayross | NULL | NULL | Bombay | Maharashtra | 400054 | 15000.00 |

| C00002 | Vandana Saitwal | NULL | NULL | Madras | Tamil Nadu | 780001 | 0.00 |

| C00003 | Pramada Jaguste | NULL | NULL | Bombay | Maharashtra | 400057 | 5000.00 |

| C00004 | Basu Navindgi | NULL | NULL | Bombay | Maharashtra | 400056 | 0.00 |

| C00005 | Ravi Sreedharan | NULL | NULL | Delhi | | 100001 | 2000.00 |

| C00006 | Rukmini | NULL | NULL | Bombay | Maharashtra | 400050 | 15000.00 |

+-----------+-----------------+----------+----------+--------+-------------+---------+----------+

3) Retrieve the list of names and the cities of all the clients.

mysql> select name,city from Client\_master;

+-----------------+--------+

| name | city |

+-----------------+--------+

| Ivan Bayross | Bombay |

| Vandana Saitwal | Madras |

| Pramada Jaguste | Bombay |

| Basu Navindgi | Bombay |

| Ravi Sreedharan | Delhi |

| Rukmini | Bombay |

+-----------------+--------+

4) List the various products available from the product\_master.

mysql> select Description from product\_master3;

+---------------+

| Description |

+---------------+

| 1.44 Floppies |

| Monitors |

| Mouse |

| 1.22 Floppies |

| Keyboards |

| CD Drive |

| 540 HDD |

| 1.44 Drive |

| 1.22 Drive |

| discs |

+---------------+

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

mysql> select name from Client\_master where name like '\_a%' ;

+-----------------+

| name |

+-----------------+

| Vandana Saitwal |

| Basu Navindgi |

| Ravi Sreedharan |

+-----------------+

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

mysql> select name,city from Client\_master where city like '\_a%' ;

+-----------------+--------+

| name | city |

+-----------------+--------+

| Vandana Saitwal | Madras |

+-----------------+--------+

7) Find the list of all clients who stay in bombay or city delhi or city madras.

mysql> select name,city from Client\_master where city="Bombay" OR city="Delhi" OR

city="Madras";

+-----------------+--------+

| name | city |

+-----------------+--------+

| Ivan Bayross | Bombay |

| Vandana Saitwal | Madras |

| Pramada Jaguste | Bombay |

| Basu Navindgi | Bombay |

| Ravi Sreedharan | Delhi |

| Rukmini | Bombay |

+-----------------+--------+

8) List all the clients who are located in 'Bombay'.

mysql> select name,city from Client\_master where city="Bombay" ;

+-----------------+--------+

| name | city |

+-----------------+--------+

| Ivan Bayross | Bombay |

| Pramada Jaguste | Bombay |

| Basu Navindgi | Bombay |

| Rukmini | Bombay |

+-----------------+--------+

9) Print the list of clients whose bal\_due are greater than value 10000

mysql> select name, Bal\_due from Client\_master where Bal\_due>10000;

+--------------+----------+

| name | Bal\_due |

+--------------+----------+

| Ivan Bayross | 15000.00 |

| Rukmini | 15000.00 |

+--------------+----------+

10) Print the information from sales\_order table of order placed in month of january.

mysql> select \* from sales\_order43 where (S\_order\_date LIKE '\_\_\_\_-01-\_\_');

+------------+--------------+-----------+-----------+-------------+-----------+-----------+------------+--------------+

| S\_order\_no | S\_order\_date | Client\_no | Dely\_addr | Salesman\_no | Dely\_type | Billed\_yn | Dely\_date | Order\_status |

+------------+--------------+-----------+-----------+-------------+-----------+-----------+------------+--------------+

| O19001 | 1996-01-12 | C00001 | NULL | S00001 | F | N | 1996-01-20 | IP |

| O19002 | 1996-01-25 | C00002 | NULL | S00002 | P | N | 1996-01-27 | C |

+------------+--------------+-----------+-----------+-------------+-----------+-----------+------------+--------------+

11) Display order information for client\_no 'c00001' and 'c00002' ;

mysql> select \* from Client\_master where Client\_no='C00001' OR

Client\_no='C00002';

+-----------+-----------------+----------+----------+--------+-------------+---------+----------+

| Client\_no | Name | Address1 | Address2 | City | State | Pincode | Bal\_due |

+-----------+-----------------+----------+----------+--------+-------------+---------+----------+

| C00001 | Ivan Bayross | NULL | NULL | Bombay | Maharashtra | 400054 | 15000.00 |

| C00002 | Vandana Saitwal | NULL | NULL | Madras | Tamil Nadu | 780001 | 0.00 |

+-----------+-----------------+----------+----------+--------+-------------+---------+----------+

12) Find the products with description as '1.44 drive' and '1.22 drive' .

mysql> select Product\_no, Description from product\_masterwhere Description='1.44

Drive' or Description='1.22 Drive';

+------------+-------------+

| Product\_no | Description |

+------------+-------------+

| P07975 | 1.44 Drive |

| P08865 | 1.22 Drive |

+------------+-------------+

13) Find the product whose selling price is greater than 2000 and less than or equal to 5000

mysql> select Product\_no,Description,Sell\_price from product\_masterwhere

Sell\_price BETWEEN 2000 AND 5000;

+------------+-------------+------------+

| Product\_no | Description | Sell\_price |

+------------+-------------+------------+

| P07868 | Keyboards | 3150.00 |

+------------+-------------+------------+

14) Find the product whose selling price is more than 1500 and also find the new selling price as

original price \* 15

mysql> select Product\_no,Description,Sell\_price, Sell\_price\*15 as original\_price

from product\_masterwhere Sell\_price >1500;

+------------+-------------+------------+----------------+

| Product\_no | Description | Sell\_price | original\_price |

+------------+-------------+------------+----------------+

| P03453 | Monitors | 12000.00 | 180000.00 |

| P07868 | Keyboards | 3150.00 | 47250.00 |

| P07885 | CD Drive | 5250.00 | 78750.00 |

| P07965 | HDD | 8400.00 | 126000.00 |

+------------+-------------+------------+----------------+

15) Rename the new in the above query as new\_price

mysql> select Product\_no,Description,Sell\_price, Sell\_price\*15 as new\_price from

product\_masterwhere Sell\_price >1500;

+------------+-------------+------------+-----------+

| Product\_no | Description | Sell\_price | new\_price |

+------------+-------------+------------+-----------+

| P03453 | Monitors | 12000.00 | 180000.00 |

| P07868 | Keyboards | 3150.00 | 47250.00 |

| P07885 | CD Drive | 5250.00 | 78750.00 |

| P07965 | HDD | 8400.00 | 126000.00 |

+------------+-------------+------------+-----------+

16) Find the product whose cost price is less than 1500

mysql> select Product\_no,Description,Cost\_price from product\_masterwhere

Cost\_price<1500;

+------------+---------------+------------+

| Product\_no | Description | Cost\_price |

+------------+---------------+------------+

| P00001 | 1.44 Floppies | 500.00 |

| P06734 | Mouse | 1000.00 |

| P07865 | 1.22 Floppies | 500.00 |

| P07975 | 1.44 Drive | 1000.00 |

| P08865 | 1.22 Drive | 1000.00 |

+------------+---------------+------------+

17) List the product in sorted order of their description

mysql>select Description from product\_masterORDER BY Description ASC;

+---------------+

| Description |

+---------------+

| 1.22 Drive |

| 1.22 Floppies |

| 1.44 Drive |

| 1.44 Floppies |

| CD Drive |

| HDD |

| Keyboards |

| Monitors |

| Mouse |

+---------------+

18) Calculate the square root of price of each product.

mysql> select Product\_no,Description,Cost\_price,sqrt(Cost\_price) as

square\_root\_of\_cost\_price from product\_master3;

+------------+---------------+------------+---------------------------+

| Product\_no | Description | Cost\_price | square\_root\_of\_cost\_price |

+------------+---------------+------------+---------------------------+

| P00001 | 1.44 Floppies | 500.00 | 22.360679774997898 |

| P03453 | Monitors | 11280.00 | 106.20734437881403 |

| P06734 | Mouse | 1000.00 | 31.622776601683793 |

| P07865 | 1.22 Floppies | 500.00 | 22.360679774997898 |

| P07868 | Keyboards | 3050.00 | 55.226805085936306 |

| P07885 | CD Drive | 5100.00 | 71.4142842854285 |

| P07965 | HDD | 8000.00 | 89.44271909999159 |

| P07975 | 1.44 Drive | 1000.00 | 31.622776601683793 |

| P08865 | 1.22 Drive | 1000.00 | 31.622776601683793 |

+------------+---------------+------------+---------------------------+

19) Divide the cost of product '540 HDD' by difference between its price and 100.

select Cost\_price/(Cost\_price-100) as Difference from product\_masterwhere

Description = 'HDD' ;

+------------+

| Difference |

+------------+

| 1.012658 |

+------------+

20) List the names,city,state of clients not in the state of 'Maharashtra' .

mysql>select name,city,state from Client\_master where NOT state ='Maharashtra';

+-----------------+--------+------------+

| name | city | state |

+-----------------+--------+------------+

| Vandana Saitwal | Madras | Tamil Nadu |

| Ravi Sreedharan | Delhi | |

+-----------------+--------+------------+

21) List the product\_no,description,sell\_price of products whose description begin with letter 'M' .

select Product\_no,Description,Sell\_price from product\_masterwhere Description

like 'M%';

+------------+-------------+------------+

| Product\_no | Description | Sell\_price |

+------------+-------------+------------+

| P03453 | Monitors | 12000.00 |

| P06734 | Mouse | 1050.00 |

+------------+-------------+------------+

22) List of all orders that were cancelled in month of March.

select \* from sales\_order43 where Order\_status='C' AND S\_order\_date BETWEEN '1996-03-01' AND

'1996-03-31';

Empty set

1) Print the description and total qty sold for each product.

mysql> select s.Product\_no,p.Description, sum(s.Qty\_ordered) from

sales\_order\_details43 s,product\_masterp where p.Product\_no=s.Product\_no group by

s.Product\_no,p.Description;

+------------+---------------+--------------------+

| Product\_no | Description | sum(s.Qty\_ordered) |

+------------+---------------+--------------------+

| P00001 | 1.44 Floppies | 34 |

| P03453 | Monitors | 6 |

| P06734 | Mouse | 1 |

| P07868 | Keyboards | 3 |

| P07885 | CD Drive | 5 |

| P07965 | HDD | 3 |

| P07975 | 1.44 Drive | 6 |

+------------+---------------+--------------------+

2) Find the value of each product sold.

mysql> select s.Product\_no,p.Description,sum(s.Qty\_disp\*s.Product\_rate) "Sales Per

Product" from sales\_order\_details43 s,product\_masterp where

p.Product\_no=s.Product\_no group by s.Product\_no,p.Description;

+------------+---------------+-------------------+

| Product\_no | Description | Sales Per Product |

+------------+---------------+-------------------+

| P00001 | 1.44 Floppies | 9975.00 |

| P03453 | Monitors | 6300.00 |

| P06734 | Mouse | 12000.00 |

| P07868 | Keyboards | 9450.00 |

| P07885 | CD Drive | 10500.00 |

| P07965 | HDD | 8400.00 |

| P07975 | 1.44 Drive | 3150.00 |

1) Find the product\_no and description of non-moving products (eg. products not being sold).

mysql> select Product\_no,Description from product\_masterwhere Product\_no not

in(select Product\_no from sales\_order\_details43);

+------------+---------------+

| Product\_no | Description |

+------------+---------------+

| P07865 | 1.22 Floppies |

| P08865 | 1.22 Drive |

+------------+---------------+

2) Find the customers name ,city and pincode for the client who has placed order no "O19001'.

mysql> select Name,Address1,Address2,City,Pincode from Client\_master where

Client\_no in (select Client\_no from sales\_order43 where S\_order\_no='O19001');

+--------------+----------+----------+--------+---------+

| Name | Address1 | Address2 | City | Pincode |

+--------------+----------+----------+--------+---------+

| Ivan Bayross | NULL | NULL | Bombay | 400054 |

+--------------+----------+----------+--------+---------+

**Views:**

Views in SQL are considered as a virtual table. A view also contains rows and columns. To create the view, we can select the fields from one or more tables present in the database. A view can either have specific rows based on certain condition or all the rows of a table. A view can be created using the CREATE VIEW statement. We can create a view from a single table or multiple tables.

**Syntax:**

CREATE VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE condition;

Just like table query, we can query the view to view the data:

SELECT \* FROM view\_name;

Examples on Creating Views :

1. Creating a view from single table –

View name – V1

CREATE VIEW V1

AS SELECT client\_no,name,city

FROM Client\_master;

SELECT \* FROM V1;

**Triggers:**

A trigger is a stored procedure in database which automatically invokes whenever a special event in the

database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when

certain table columns are being updated.

**Syntax:**

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

[trigger\_body]

**Examples of creating Triggers -**

**1. Upper Case**

Step 1. Create table ‘student’ :

mysql> create table Student

(

roll\_no int(2),

name varchar(20),

city varchar(15),

state varchar(20)

);

**Step 2. Create a trigger** :

mysql> delimiter $$

create trigger tg1

before insert on Student

for each row

begin

set new.name= upper(new.name);

set new.city= upper(new.city);

end;

$$

Step 3. Insert values into student table :

mysql> insert into Student values (28, 'Shania', 'mumbai', 'Maharashtra');

-> $$

mysql> insert into Student values (36, 'Simran', 'ahmedabad', 'Gujrat');

-> $$

mysql> insert into Student values (12, 'Neha', 'gangtok', 'Sikkim');

-> $$

Step 4. View the student table :

mysql> select \* from Student;

-> $$

+---------+--------+-----------+-------------+

| roll\_no | name | city | state |

+---------+--------+-----------+-------------+

| 28 | SHANIA | MUMBAI | Maharashtra |

| 36 | SIMRAN | AHMEDABAD | Gujrat |

| 12 | NEHA | GANGTOK | Sikkim |

+---------+--------+-----------+-------------+

**2. After Updation**

- Step 1. Create table ‘person’ :

mysql> create table person

(

fname char(10),

lname char(10),

id decimal(8) primary key

);

$$

- Step 2. Insert values in person table :

mysql> insert into person values ('leena','kirtikar',43);

-> $$

mysql> insert into person values ('aakash','kapadia',38);

-> $$

mysql> insert into person values ('parth','shroff',63);

-> $$

mysql> select \* from person;

-> $$

+--------+----------+----+

| fname | lname | id |

+--------+----------+----+

| aakash | kapadia | 38 |

| leena | kirtikar | 43 |

| parth | shroff | 63 |

+--------+----------+----+

-Step 3. Create table ‘audit\_log’ :

mysql> create table audit\_log

(

ofname char(10),

olname char(10),

nfname char(10),

nname char(10),

cwhen date

);

$$

Step 4. Create a trigger :

mysql> delimiter $$

create trigger t2

after update on person

for each row

begin

insert into audit\_log

values(old.fname,old.lname,new.fname,new.lname,curdate()

);

end;

$$

Step 5. Work on trigger using the queries :

mysql> update person set fname='anish' where id like 43;

-> $$

mysql> update person set fname='anvay' where id like 63;

-> $$

mysql> update person set fname='aman' where id like 38;

-> $$

Step 6. View both the tables :

mysql> select \*from audit\_log;

-> $$

+--------+----------+--------+----------+------------+

| ofname | olname | nfname | nname | cwhen |

+--------+----------+--------+----------+------------+

| leena | kirtikar | anish | kirtikar | 2020-11-26 |

| parth | shroff | anvay | shroff | 2020-11-26 |

| aakash | kapadia | aman | kapadia | 2020-11-26 |

+--------+----------+--------+----------+------------+

**Procedures:**

A procedure is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

mysql> delimiter &&

create procedure my\_pro(IN client\_no int)

begin

select \* from Client\_master LIMIT client\_no ;

end;

&&

mysql> call my\_pro(2);

&&

+-----------+-----------------+----------+----------+--------+-------------+---------+---------+

| Client\_no | Name | Address1 | Address2 | City | State | Pincode | Bal\_due |

+-----------+-----------------+----------+----------+--------+-------------+---------+---------+

| C00001 | Ivan Bayross | NULL | NULL | Bombay | Maharashtra | 400054 | 1000.00 |

| C00002 | Vandana Saitwal | NULL | NULL | Madras | Tamil Nadu | 780001 | 0.00 |

+-----------+-----------------+----------+----------+--------+-------------+---------+---------

mysql> delimiter &&

create procedure my\_pro1(OUT client\_no int)

begin

select COUNT(\*) INTO client\_no FROM Client\_master;

end;

&&

call my\_pro1(@a);

&&

select @a;

&&

+------+

| @a |

+------+

| 6 |

+------+

**Functions:**

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

CREATE [OR REPLACE] FUNCTION function\_name [parameters]

RETURN return\_datatype; {IS, AS}

Declaration\_section <variable,constant> ;

BEGIN Execution\_section

Return return\_variable;

EXCEPTION exception section

Return return\_variable;

END;

mysql> delimiter $$

create function proprice(sell\_price double)RETURNS varchar(20)

DETERMINISTIC

BEGIN

DECLARE lvl varchar(20);

IF sell\_price<1000 THEN

SET lvl='CHEAP';

ELSEIF sell\_price>3000 THEN

SET lvl='EXPENSIVE';

END IF;

RETURN(lvl);

END;

$$

mysql> select product\_no,proprice(sell\_price) from product\_master;

$$

**Cursor:**

To handle a result set inside a stored procedure, you use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row individually.

MySQL cursor is read-only, non-scrollable and a sensitive.

**Example 1 :**

mysql> delimiter $$

CREATE procedure CPr()

BEGIN

DECLARE done INT(10) DEFAULT 0;

DECLARE i varchar(20);

DECLARE n varchar(100);

declare curs1 CURSOR FOR select roll\_no,name from Student;

declare CONTINUE HANDLER FOR NOT FOUND set done=1;

OPEN curs1;

read\_loop:LOOP

fetch curs1 INTO i,n;

If done=1 THEN

leave read\_loop;

end if;

select i,n;

end loop read\_loop;

close curs1;

end;

$$

mysql> CALL CPr();

$$

+------+--------+

| i | n |

+------+--------+

| 28 | SHANIA |

+------+--------+

+------+--------+

| i | n |

+------+--------+

| 36 | SIMRAN |

+------+--------+

+------+------+

| i | n |

+------+------+

| 12 | NEHA |

+------+------+

Example 2 :

mysql> alter table person add column email varchar(50);

-> $$

mysql> update person set email='aman@gmail.com' where id like 38;

-> update person set email='anish@gmail.com' where id like 43;

-> update person set email='anvay@gmail.com' where id like 63;

-> $$

mysql> select \* from person;

-> $$

+-------+----------+----+-----------------+

| fname | lname | id | email |

+-------+----------+----+-----------------+

| aman | kapadia | 38 | aman@gmail.com |

| anish | kirtikar | 43 | anish@gmail.com |

| anvay | shroff | 63 | anvay@gmail.com |

+-------+----------+----+-----------------+

mysql> delimiter $$

CREATE procedure build\_email\_list(INOUT email\_list varchar(400))

BEGIN

DECLARE v\_finished INTEGER DEFAULT 0;

declare v\_email varchar(100) DEFAULT "";

DECLARE email\_cursor CURSOR FOR

SELECT email from person;

DECLARE CONTINUE HANDLER

FOR NOT FOUND set v\_finished=1;

OPEN email\_cursor;

get\_email:LOOP

FETCH email\_cursor INTO v\_email;

IF v\_finished=1 THEN

LEAVE get\_email;

END IF;

SET email\_list=CONCAT(v\_email,";",email\_list);

END LOOP get\_email;

CLOSE email\_cursor;

END;

$$

SET @email\_list="";

$$

CALL build\_email\_list(@email\_list);

$$

select @email\_list;

$$

+-------------------------------------------------+

| @email\_list |

+-------------------------------------------------+

| anvay@gmail.com;anish@gmail.com;aman@gmail.com; |

+-------------------------------------------------+

SET @email\_list="2";

$$

CALL build\_email\_list(@email\_list);

$$

select @email\_list;

$$

+--------------------------------------------------+

| @email\_list |

+--------------------------------------------------+

| anvay@gmail.com;anish@gmail.com;aman@gmail.com;2 |

+--------------------------------------------------+

**Aim: - Study of Database Connectivity using JDBC**

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

 Making a connection to a database.

 Creating SQL or MySQL statements.

 Executing SQL or MySQL queries in the database.

 Viewing & Modifying the resulting records.

JDBC Architecture

The JDBC API supports both two-tier and three-tier processing models for database access but in general,

JDBC Architecture consists of two layers −

 JDBC API: This provides the application-to-JDBC Manager connection.

 JDBC Driver API: This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous

databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the

JDBC drivers and the Java application −

Common JDBC Components

The JDBC API provides the following interfaces and classes −

 DriverManager: This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.

 Driver: This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.

 Connection: This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.

 Statement: You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.

 ResultSet: These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.

 SQLException: This class handles any errors that occur in a database application.

Steps for creating Database Connectivity Applications:

There are mainly six steps:

Step1: Import the packages Required for Database Programming.

Step2: Register the JDBC Driver to JDBC Driver manager

Step3: Open a connection

Step4: Execute a Query

Step5: Extract Data from Result set

Step6: Close Connection.