BTECCE21405: Database Management Systems Lab

NAME	Atharva shevate
YEAR	SECOND YEAR
ROLL NO	02
SRN NO	202201727
DIV	E
BATCH	E1
SUBJECT	DBMS

Assignment 3
Joins, Sets, Subqueries

1. Consider following Database Schemas **Account(Acc_no, branch_name,balance)**
branch(branch_name,branch_city,assets) customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no) Solve following query:
Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc.

```
mysql> use assignment_5_a;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------------+
| Tables_in_assignment_5_a |
+-------------------------+
| account                 |
| borrower                |
| branch                  |
| customer                |
| depositor               |
| loan                    |
+-------------------------+
6 rows in set (0.01 sec)
```

Table 1: Account.

```
mysql> desc account;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| acc_no      | int         | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| balance     | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table 2: Branch.

```
mysql> desc branch;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| branch_name | varchar(20) | NO   | PRI | NULL    |       |
| branch_city | varchar(20) | YES  |     | NULL    |       |
| assets      | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table 3: Customer.

```
mysql> desc customer;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| cust_name   | varchar(50) | NO   | PRI | NULL    |       |
| cust_street | varchar(50) | YES  |     | NULL    |       |
| cust_city   | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table 4: Depositor

```
mysql> desc depositor;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| cust_name | varchar(50) | NO   | PRI | NULL    |       |
| acc_no    | int         | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> desc loan;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| loan_no     | int         | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Table 5: Loan

```
mysql> desc borrower;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| cust_name | varchar(50) | NO   | PRI | NULL    |       |
| loan_no   | int         | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Table 6: Borrower

Record 1:Account

```
mysql> select * from account;
+--------+-------------+---------+
| acc_no | branch_name | balance |
+--------+-------------+---------+
|  10001 | Mumbai      |   10000 |
|  10002 | Delhi       |   15000 |
|  10003 | Kolkata     |    5000 |
|  10004 | Chennai     |   23000 |
|  10005 | Bangalore   |    3000 |
|  10006 | Hyderabad   |    8000 |
|  10007 | Pune        |   12000 |
|  10008 | Ahmedabad   |    9000 |
|  10009 | Jaipur      |   18000 |
|  10010 | Chandigarh  |    7500 |
|  10011 | Pune        |   11500 |
|  10012 | Bangalore   |    2200 |
|  10013 | Jaipur      |   12200 |
|  10014 | Chennai     |   17050 |
|  10015 | Pune        |   22550 |
|  10016 | Chandigarh  |     750 |
|  10017 | Ahmedabad   |   11750 |
|  10018 | Kolkata     |    2100 |
|  10019 | Mumbai      |   12100 |
|  10020 | Delhi       |    3400 |
+--------+-------------+---------+
20 rows in set (0.01 sec)
```

Record 2: Branch.

```
mysql> select * from branch;
+-------------+-------------+--------+
| branch_name | branch_city | assets |
+-------------+-------------+--------+
| Ahmedabad   | Ahmedabad   |  70000 |
| Bangalore   | Bangalore   |  60000 |
| Chandigarh  | Chandigarh  |  50000 |
| Chennai     | Chennai     | 200000 |
| Delhi       | Delhi       | 150000 |
| Hyderabad   | Hyderabad   | 100000 |
| Jaipur      | Jaipur      | 110000 |
| Kolkata     | Kolkata     |  80000 |
| Mumbai      | Mumbai      | 120000 |
| Pune        | Pune        |  90000 |
+-------------+-------------+--------+
10 rows in set (0.00 sec)
```

Record 3:Customer.

```
mysql> select * from customer;
+---------------+---------------------+------------+
| cust_name     | cust_street         | cust_city  |
+---------------+---------------------+------------+
| Deepak Singh  | 1313 Gandhi Nagar   | Bhopal     |
| Jaya Sharma   | 567 Radha Nagar     | Jaipur     |
| Manisha Reddy | 1414 Indira Street  | Hyderabad  |
| Neha Patel    | 123 Main Road       | Mumbai     |
| Nitin Joshi   | 890 Vrindavan Colony| Ahemadabad |
| Pooja Verma   | 1122 Kailash Nagar  | Chandigarh |
| Rajesh Shah   | 456 Gandhi Lane     | Delhi      |
| Renuka Singh  | 789 Sardar Street   | Kolkata    |
| Rohit Gupta   | 101 Tulsidas Marg   | Lucknow    |
| Suresh Kumar  | 234 Nehru Road      | Chennai    |
+---------------+---------------------+------------+
10 rows in set (0.00 sec)
```

Record 4:Depositor.

```
mysql> select * from depositor;
+---------------+--------+
| cust_name     | acc_no |
+---------------+--------+
| Neha Patel    |  10001 |
| Rajesh Shah   |  10002 |
| Renuka Singh  |  10003 |
| Rohit Gupta   |  10004 |
| Suresh Kumar  |  10005 |
| Jaya Sharma   |  10006 |
| Nitin Joshi   |  10007 |
| Pooja Verma   |  10008 |
| Deepak Singh  |  10009 |
| Manisha Reddy |  10010 |
| Neha Patel    |  10011 |
| Rajesh Shah   |  10012 |
| Renuka Singh  |  10013 |
| Rohit Gupta   |  10014 |
| Suresh Kumar  |  10015 |
| Jaya Sharma   |  10016 |
| Nitin Joshi   |  10017 |
| Pooja Verma   |  10018 |
| Deepak Singh  |  10019 |
| Manisha Reddy |  10020 |
+---------------+--------+
20 rows in set (0.01 sec)
```

Record 5: Loan.

```
mysql> select * from loan;
+---------+-------------+--------+
| loan_no | branch_name | amount |
+---------+-------------+--------+
|    1001 | Mumbai      |  15000 |
|    1002 | Delhi       |  20000 |
|    1003 | Bangalore   |  10000 |
|    1004 | Hyderabad   |  25000 |
|    1005 | Kolkata     |  30000 |
|    1006 | Mumbai      |  12000 |
|    1007 | Bangalore   |  18000 |
|    1008 | Chennai     |  50000 |
|    1009 | Delhi       |  40000 |
|    1010 | Pune        |  22000 |
+---------+-------------+--------+
10 rows in set (0.00 sec)
```

Record 6: Borrower.

```
mysql> select * from borrower;
+---------------+---------+
| cust_name     | loan_no |
+---------------+---------+
| Manisha Reddy |    1001 |
| Pooja Verma   |    1002 |
| Rohit Gupta   |    1003 |
| Jaya Sharma   |    1004 |
| Nitin Joshi   |    1005 |
| Renuka Singh  |    1006 |
| Suresh Kumar  |    1007 |
| Deepak Singh  |    1008 |
| Rajesh Shah   |    1009 |
| Neha Patel    |    1010 |
+---------------+---------+
10 rows in set (0.00 sec)
```

1) Find the names of all branches in loan relation.

```
mysql> select distinct branch_name from loan;
+-------------+
| branch_name |
+-------------+
| Bangalore   |
| Chennai     |
| Delhi       |
| Hyderabad   |
| Kolkata     |
| Mumbai      |
| Pune        |
+-------------+
7 rows in set (0.00 sec)
```

2) A)Find all loan numbers for loans made at Mumbai Branch with loan amount > 12000.

```
mysql> select loan_no from loan where branch_name= 'Mumbai' AND amou
nt > 12000;
+---------+
| loan_no |
+---------+
|    1001 |
+---------+
1 row in set (0.00 sec)
```

B)Find all loan numbers for loans made at Mumbai Branch with loan amount >
=12000.

```
mysql> select loan_no from loan where branch_name= 'Mumbai' AND amou
nt >= 12000;
+---------+
| loan_no |
+---------+
|    1001 |
|    1006 |
+---------+
2 rows in set (0.00 sec)
```

3) Find all customers who have a loan from bank. Find their names,loan_no and loan amount.

```
mysql> select c.cust_name, l.loan_no , l.amount
    -> from customer c, loan l, borrower b
    -> where c.cust_name = b.cust_name AND b.loan_no = l.loan_no;
+---------------+---------+--------+
| cust_name     | loan_no | amount |
+---------------+---------+--------+
| Deepak Singh  |    1008 |  50000 |
| Jaya Sharma   |    1004 |  25000 |
| Manisha Reddy |    1001 |  15000 |
| Neha Patel    |    1010 |  22000 |
| Nitin Joshi   |    1005 |  30000 |
| Pooja Verma   |    1002 |  20000 |
| Rajesh Shah   |    1009 |  40000 |
| Renuka Singh  |    1006 |  12000 |
| Rohit Gupta   |    1003 |  10000 |
| Suresh Kumar  |    1007 |  18000 |
+---------------+---------+--------+
10 rows in set (0.01 sec)
```

4) List all customers in alphabetical order who have loan from Mumbai branch.

```
mysql> select c.cust_name
    -> from customer c , loan l, borrower b
    -> where c.cust_name = b.cust_name AND b.loan_no=l.loan_no AND
    -> l.branch_name='Mumbai'
    -> Order by c.cust_name ASC;
+---------------+
| cust_name     |
+---------------+
| Manisha Reddy |
| Renuka Singh  |
+---------------+
2 rows in set (0.00 sec)
```

5) Find all customers who have an account or loan or both at bank.

```
mysql> select distinct d.cust_name from depositor d left join custo
mer c on d.cust_name = c.cust_name left join account a on d.acc_no
=
a.acc_no;
+---------------+
| cust_name     |
+---------------+
| Neha Patel    |
| Rajesh Shah   |
| Renuka Singh  |
| Rohit Gupta   |
| Suresh Kumar  |
| Jaya Sharma   |
| Nitin Joshi   |
| Pooja Verma   |
| Deepak Singh  |
| Manisha Reddy |
+---------------+
10 rows in set (0.00 sec)
```

6) Find all customers who have both account and loan at bank.

```
mysql> select distinct d.cust_name
    -> from depositor d
    -> inner join borrower b on d.cust_name = b.cust_name
    -> inner join account a on d.acc_no = a.acc_no;
+---------------+
| cust_name     |
+---------------+
| Manisha Reddy |
| Pooja Verma   |
| Rohit Gupta   |
| Jaya Sharma   |
| Nitin Joshi   |
| Renuka Singh  |
| Suresh Kumar  |
| Deepak Singh  |
| Rajesh Shah   |
| Neha Patel    |
+---------------+
10 rows in set (0.00 sec)
```

7) Find all customer who have account but no loan at the bank.

```
mysql> select distinct d.cust_name from depositor d left join
borrower b on d.cust_name = b.cust_name where b.cust_name is n
ull;
Empty set (0.01 sec)
```

8) Find average account balance at Mumbai branch.

```
mysql> select avg(balance)
    -> from account
    -> where branch_name = 'Mumbai';
+--------------+
| avg(balance) |
+--------------+
|   11050.0000 |
+--------------+
1 row in set (0.00 sec)
```

9. Find the average account balance at each branch

```
mysql> select branch_name, avg(balance)
    -> from account
    -> group by branch_name;
+-------------+--------------+
| branch_name | avg(balance) |
+-------------+--------------+
| Ahmedabad   |   10375.0000 |
| Bangalore   |    2600.0000 |
| Chandigarh  |    4125.0000 |
| Chennai     |   20025.0000 |
| Delhi       |    9200.0000 |
| Hyderabad   |    8000.0000 |
| Jaipur      |   15100.0000 |
| Kolkata     |    3550.0000 |
| Mumbai      |   11050.0000 |
| Pune        |   15350.0000 |
+-------------+--------------+
10 rows in set (0.01 sec)
```

10) Find no. of depositors at each branch.

```
mysql> select count(distinct d.cust_name) from depositor d left
t join account a on d.acc_no = a.acc_no group by branch_name;
+----------------------------+
| count(distinct d.cust_name) |
+----------------------------+
|                          2 |
|                          2 |
|                          2 |
|                          1 |
|                          2 |
|                          1 |
|                          2 |
|                          2 |
|                          2 |
|                          3 |
+----------------------------+
10 rows in set (0.01 sec)
```

11) Find the branches where average account balance > 12000.

```
mysql> select branch_name
    -> from account
    -> group by branch_name
    -> having avg(balance)> 12000;
+-------------+
| branch_name |
+-------------+
| Chennai     |
| Jaipur      |
| Pune        |
+-------------+
3 rows in set (0.00 sec)
```

12) Find number of tuples in customer relation.

```
mysql> select count(*)
    -> from customer;
+----------+
| count(*) |
+----------+
|       10 |
+----------+
1 row in set (0.01 sec)
```

13) Calculate total loan amount given by bank.

```
mysql> select sum(amount)
    -> from loan;
+-------------+
| sum(amount) |
+-------------+
|      242000 |
+-------------+
1 row in set (0.00 sec)
```

Q2) Consider the given relational table:
employee(empno , empname, designation, city, salary, zipcode, county)

1. Creates a sequence used to generate employee numbers for the empno column of the emp table.
2. Create an Index on county.
3. Find the zipcode whose county = 071 and check whether the query uses the Index and write your observation.
4. Create a view for employees having salary < 50000 and stays in 'Mumbai'

```
mysql> use assignment_5_b;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -
A

Database changed
mysql> show tables;
+--------------------------+
| Tables_in_assignment_5_b |
+--------------------------+
| employee                 |
+--------------------------+
1 row in set (0.00 sec)
```

Table : Employee:

```
mysql> desc employee;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| amp_no      | int         | NO   | PRI | NULL    |       |
| emp_name    | varchar(30) | YES  |     | NULL    |       |
| designation | varchar(30) | YES  |     | NULL    |       |
| city        | varchar(20) | YES  |     | NULL    |       |
| salary      | int         | YES  |     | NULL    |       |
| zip_code    | int         | YES  |     | NULL    |       |
| country     | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Record : Employee.

```
mysql> select * from employee;
+--------+-----------+-------------+-----------+--------+----------+---------+
| amp_no | emp_name  | designation | city      | salary | zip_code | country |
+--------+-----------+-------------+-----------+--------+----------+---------+
|      1 | John Doe  | Manager     | Mumbai    |  75000 |   400001 |      71 |
|      2 | Jane Smith| Engineer    | Bangalore |  60000 |   560001 |      80 |
|      3 | Bob Brown | Analyst     | Chennai   |  45000 |   600001 |      44 |
|      4 | Tom Lee   | Developer   | Mumbai    |  40000 |   400005 |      71 |
|      5 | Ann Chen  | Manager     | Kolkata   |  80000 |   700001 |      33 |
|      6 | Sam Kim   | Designer    | Mumbai    |  35000 |   400007 |      71 |
+--------+-----------+-------------+-----------+--------+----------+---------+
6 rows in set (0.00 sec)
```

Queries:

1)Creates a sequence used to generate employee numbers for the empno column of the emp table.

```
CREATE SEQUENCE empno_seq START WITH 1 INCREMENT BY 1;
```

```
mysql> insert into employee
    -> values (nextval('empno_seq'), 'John Doe' , 'Manager',
'Mumbai', 75000,'400001', 071);
```

2)Create an Index on country.

```
mysql> create index idx_country on employee(country);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc employee;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| emp_no      | int         | NO   | PRI | NULL    |       |
| emp_name    | varchar(30) | YES  |     | NULL    |       |
| designation | varchar(30) | YES  |     | NULL    |       |
| city        | varchar(20) | YES  |     | NULL    |       |
| salary      | int         | YES  |     | NULL    |       |
| zip_code    | int         | YES  |     | NULL    |       |
| country     | int         | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

3)Find the zipcode whose country = 071 and check whether the query uses the Index and write your observation.

```
mysql> explain analyze
    -> select zip_code
    -> from employee
    -> where country = 071 ;
+---------------------------------------------------------------------
-------------------------------------------------------+
| EXPLAIN
                                                       |
+---------------------------------------------------------------------
-------------------------------------------------------+
| -> Index lookup on employee using idx_country (country=71)  (cost=0
.80 rows=3) (actual time=0.068..0.082 rows=3 loops=1)
 |
+---------------------------------------------------------------------
-------------------------------------------------------+
1 row in set (0.01 sec)
```

4)Create a view for employees having salary < 50000 and stays in 'Mumbai'.

```
mysql> select * from low_salary_mumbai_employees;
+----------+-------------+--------+--------+----------+---------+
| emp_name | designation | city   | salary | zip_code | country |
+----------+-------------+--------+--------+----------+---------+
| Tom Lee  | Developer   | Mumbai |  40000 |   400005 |      71 |
| Sam Kim  | Designer    | Mumbai |  35000 |   400007 |      71 |
+----------+-------------+--------+--------+----------+---------+
2 rows in set (0.00 sec)
```

```
mysql> show create view low_salary_mumbai_employee;
+-----------------------------+---------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------+------------------------------+-
---------------------+
| View                        | Create View

                                                                                                | character_set_client |
collation_connection |
+-----------------------------+---------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------+------------------------------+-
---------------------+
| low_salary_mumbai_employee | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `low_sa
lary_mumbai_employee` AS select `employee`.`emp_name` AS `emp_name`,`employee`.`designation` AS `designation`,`employe
e`.`city` AS `city`,`employee`.`salary` AS `salary`,`employee`.`zip_code` AS `zip_code`,`employee`.`country` AS `count
ry` from `employee` where ((`employee`.`salary` < 50000) and (`employee`.`city` = 'Mumbai')) | utf8mb4              |
utf8mb4_0900_ai_ci   |
+-----------------------------+---------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------+------------------------------+-
---------------------+
1 row in set (0.00 sec)
```

Q3) Consider the given database schema:

Student (studentid , studentname,instructorid,studentcity)

Instructor(instructorid,Instructorname,instructorcity,specialization) Use all types of Joins

1. Find the instructor of each student.
2. Find the student who is not having any instructor.
3. Find the student who is not having any instructor as well as instructor who is not having student.
4. Find the students whose instructor's specialization is computer.
5. Create a view containing total number of students whose instructor belongs to "Pune".

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| assignment_2       |
| assignment_5_a     |
| assignment_5_b     |
| assignment_5_c     |
| cricket            |
| information_schema |
| mysql              |
| paper              |
| performance_schema |
| sys                |
+--------------------+
10 rows in set (0.01 sec)
```

Table 1: Student.

```
mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| student_id   | int         | NO   | PRI | NULL    |       |
| student_name | varchar(50) | YES  |     | NULL    |       |
| instructor_id| int         | YES  |     | NULL    |       |
| student_city | varchar(30) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Table 2:
Instructor.

```
mysql> desc instructor;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| instructor_id   | int         | NO   | PRI | NULL    |       |
| instructor_name | varchar(50) | YES  |     | NULL    |       |
| instructor_city | varchar(30) | YES  |     | NULL    |       |
| specialization  | varchar(30) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Record 1:
Student.

```
mysql> select * from student;
+------------+--------------+---------------+--------------+
| student_id | student_name | instructor_id | student_city |
+------------+--------------+---------------+--------------+
|          1 | Aniket       |             2 | Pune         |
|          2 | Ganesh       |             5 | Mumbai       |
|          3 | Siddhi       |             1 | Solpaur      |
|          4 | Rutuja       |             3 | Osmanabad    |
|          5 | Arnav        |             4 | Pune         |
|          6 | Anuja        |             2 | Kurduwadi    |
|          7 | Akshay       |             4 | Aurangabad   |
|          8 | Arjav        |          NULL | Pune         |
+------------+--------------+---------------+--------------+
8 rows in set (0.00 sec)
```

Record 2: Instructor.

```
mysql> select * from instructor;
+---------------+-----------------+-----------------+-------------------+
| instructor_id | instructor_name | instructor_city | specialization    |
+---------------+-----------------+-----------------+-------------------+
|             1 | Vipul Kondekar  | Solapur         | Electronics       |
|             2 | Jamel Ansari    | Pune            | Math              |
|             3 | Bashithi        | Osmanabad       | Biology           |
|             4 | Jagdesh Tawde   | Pune            | Math              |
|             5 | Sailesh Thavre  | Mumbai          | Computer          |
|             6 | Madhavi Dachwar | Mumbai          | Computer Graphics |
+---------------+-----------------+-----------------+-------------------+
6 rows in set (0.00 sec)
```

Queries:

1) Find the instructor of each student.

```
mysql> select s.student_name, i.instructor_name
    -> from student s
    -> inner join instructor i on s.instructor_id = i.instructor_id;
+--------------+-----------------+
| student_name | instructor_name |
+--------------+-----------------+
| Aniket       | Jamel Ansari    |
| Ganesh       | Sailesh Thavre  |
| Siddhi       | Vipul Kondekar  |
| Rutuja       | Bashithi        |
| Arnav        | Jagdesh Tawde   |
| Anuja        | Jamel Ansari    |
| Akshay       | Jagdesh Tawde   |
+--------------+-----------------+
7 rows in set (0.00 sec)
```

2) Find the student who is not having any instructor.

```
mysql> select s.student_name
    -> from student s
    -> left join instructor i on s.instructor_id=i.instructor_id
    -> where i.instructor_id is null;
+--------------+
| student_name |
+--------------+
| Arjav        |
+--------------+
1 row in set (0.00 sec)
```

3) Find the student who is not having any instructor as well as instructor who is not having student.

```
mysql> select s.student_name, i.instructor_name
    -> from student s
    -> full outer join instructor i on s.instructor_id= i.instrucctor_id
    -> where s.student_id is null or i.instructor_id is null;
```

4) Find the students whose instructor's specialization is computer.

```
mysql> select s.student_name, i.instructor_name
    -> from student s
    -> inner join instructor i on s.instructor_id=i.instructor_id
    -> where i.specialization = 'computer';
+--------------+-----------------+
| student_name | instructor_name |
+--------------+-----------------+
| Ganesh       | Sailesh Thavre  |
+--------------+-----------------+
1 row in set (0.00 sec)
```

5) Create a view containing total number of students whose instructor belongs to “Pune”.

```
mysql> create view pune_students as
    -> select i.instructor_city, count(s.student_id) as total_students
    -> from student s
    -> inner join instructor i on s.instructor = i.instructor_id
    -> where i.instructor_city = 'Pune'
    -> group by i.instructor_city;
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+--------------------------+
| Tables_in_assignment_5_c |
+--------------------------+
| instructor               |
| pune_students            |
| student                  |
+--------------------------+
3 rows in set (0.00 sec)

mysql> select * from pune_students;
+-----------------+----------------+
| instructor_city | total_students |
+-----------------+----------------+
| Pune            |              4 |
+-----------------+----------------+
1 row in set (0.00 sec)
```

```
mysql> show create view pune_students;
+---------------+------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
--------------------------------------------------------------------+-------
---------------+----------------------+
| View          | Create View



                                                                    | charac
ter_set_client | collation_connection |
+---------------+------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
--------------------------------------------------------------------+-------
---------------+----------------------+
| pune_students | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL 
SECURITY DEFINER VIEW `pune_students` AS select `i`.`instructor_city` AS `in
structor_city`,count(`s`.`student_id`) AS `total_students` from (`student` `
s` join `instructor` `i` on((`s`.`instructor_id` = `i`.`instructor_id`))) wh
ere (`i`.`instructor_city` = 'Pune') group by `i`.`instructor_city` | utf8mb
4              | utf8mb4_0900_ai_ci   |
+---------------+------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
--------------------------------------------------------------------+-------
---------------+----------------------+
1 row in set (0.00 sec)
```

Q4) Create a database with following schemas

Borrower(Rollin, Name, DateofIssue, NameofBook, Status) & Fine(Roll_no,Date,Amt)

1. Write a PL/SQL block to accept input for Borrower table.
2. Write a PL/SQL block using control structures to calculate fine by using the following rules:

. a. check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.

. b. If no. of days>30, per day fine will be Rs 50 per day.

. c. for days less than 30, Rs. 5 per day.

After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| assignment_2       |
| assignment_5_a     |
| assignment_5_b     |
| assignment_5_c     |
| assignment_5_d     |
| cricket            |
| information_schema |
| mysql              |
| paper              |
| performance_schema |
| sys                |
+--------------------+
11 rows in set (0.00 sec)
```

```
mysql> show tables;
+-------------------------+
| Tables_in_assignment_5_d |
+-------------------------+
| borrower                |
| fine                    |
+-------------------------+
2 rows in set (0.00 sec)
```

Table 1: Borrower.

```
mysql> desc borrower;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| roll_no       | int         | NO   | PRI | NULL    |       |
| name          | varchar(50) | YES  |     | NULL    |       |
| date_of_issue | date        | YES  |     | NULL    |       |
| name_of_book  | varchar(50) | YES  |     | NULL    |       |
| status        | varchar(10) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Table 2: Fine.

```
mysql> desc fine;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| roll_no | int  | YES  |     | NULL    |       |
| date    | date | YES  |     | NULL    |       |
| amt     | int  | YES  |     | NULL    |       |
+---------+------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Record 1:

```
mysql> select * from borrower;
+---------+-------------+---------------+------------------------+--------+
| roll_no | name        | date_of_issue | name_of_book           | status |
+---------+-------------+---------------+------------------------+--------+
|       1 | John Doe    | 2022-02-01    | The Great Gatsby       | I      |
|       2 | Jane Smith  | 2022-03-15    | To Kill a Mockingbird  | I      |
|       3 | Bob Johnson | 2022-04-01    | Pride and Prejudice    | R      |
|       4 | Sarah Lee   | 2022-05-01    | The Catcher in the Rye | I      |
+---------+-------------+---------------+------------------------+--------+
4 rows in set (0.00 sec)
```

Record 2:

```
mysql> select * from fine;
+---------+------------+------+
| roll_no | date       | amt  |
+---------+------------+------+
|       1 | 2022-03-01 |    0 |
|       2 | 2022-04-01 |    0 |
|       4 | 2022-05-20 |  250 |
+---------+------------+------+
3 rows in set (0.00 sec)
```

Queries:

1) Write a PL/SQL block to accept input for Borrower table.

```sql
CREATE TABLE Borrower (
  Rollno INTEGER PRIMARY KEY,
  Name VARCHAR(50),
  DateofIssue DATE,
  NameofBook VARCHAR(50),
  Status CHAR(1)
);

CREATE TABLE Fine (
  Roll_no INTEGER REFERENCES Borrower(Rollno),
  Date DATE,
  Amt INTEGER
);
```

2) Write a PL/SQL block using control structures to calculate fine by using the following rules:

a. check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.

b. If no. of days>30, per day fine will be Rs 50 per day.

c. for days less than 30, Rs. 5 per day.

After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.

```
DECLARE
  v_Rollno INTEGER := &Enter_Rollno;
  v_DateofIssue DATE;
  v_Status CHAR(1);
  v_FineAmt INTEGER := 0;
BEGIN
  SELECT DateofIssue, Status INTO v_DateofIssue, v_Status
  FROM Borrower
  WHERE Rollno = v_Rollno;

  IF v_Status = 'I' THEN
    IF SYSDATE - v_DateofIssue > 30 THEN
      v_FineAmt := (SYSDATE - v_DateofIssue - 30) * 50;
    ELSE
      v_FineAmt := 0;
      IF SYSDATE - v_DateofIssue > 15 THEN
        v_FineAmt := (SYSDATE - v_DateofIssue - 15) * 5;
      END IF;
    END IF;

    IF v_FineAmt > 0 THEN
      INSERT INTO Fine VALUES (v_Rollno, SYSDATE, v_FineAmt);
    END IF;

    UPDATE Borrower SET Status = 'R' WHERE Rollno = v_Rollno;
  END IF;
END;
```

Create

Q5) two tables
O_Roll(Rollno,Name,DOB,Phone,address)
N_Roll(Rollno,Name,DOB,Phone,address)

Write a PL/SQL block using various types of cursor(implicit,Explicit,For, Parameterized) to merge records from O_Roll table with that of N_Roll in such a way duplicate records are to be eliminated.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| assignment_2       |
| assignment_5_a     |
| assignment_5_b     |
| assignment_5_c     |
| assignment_5_d     |
| assignment_5_e     |
| cricket            |
| information_schema |
| mysql              |
| paper              |
| performance_schema |
| sys                |
+--------------------+
12 rows in set (0.00 sec)
```

```
mysql> select * from o_roll
    -> ;
+---------+------+------------+------------+-------------+
| roll_no | name | dob        | phone      | address     |
+---------+------+------------+------------+-------------+
|       1 | John | 1990-01-01 | 1234567890 | 123 Main St |
|       2 | Jane | 1991-02-02 | 2345678901 | 456 Elm St  |
|       3 | Jim  | 1992-03-03 | 3456789012 | 789 Oak St  |
+---------+------+------------+------------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from n_roll;
+---------+------+------------+------------+------------+
| roll_no | name | dob        | phone      | address    |
+---------+------+------------+------------+------------+
|       2 | Jane | 1991-02-02 | 2345678901 | 456 Elm St |
|       3 | Jim  | 1992-03-03 | 3456789012 | 789 Oak St |
|       4 | Jack | 1993-04-04 | 4567890123 | 12 Pine St |
+---------+------+------------+------------+------------+
3 rows in set (0.00 sec)
```

```
-- PL/SQL block
DECLARE
  CURSOR o_cursor IS
    SELECT *
    FROM O_Roll;

  n_record N_Roll%ROWTYPE;

  CURSOR n_cursor (p_rollno N_Roll.Rollno%TYPE) IS
    SELECT *
    FROM N_Roll
    WHERE Rollno = p_rollno;

  p_rollno O_Roll.Rollno%TYPE;
BEGIN
  -- implicit cursor
  FOR o_record IN o_cursor LOOP
    -- explicit cursor
    OPEN n_cursor(o_record.Rollno);
    FETCH n_cursor INTO n_record;

    -- for loop cursor
    FOR i IN 1..1 LOOP
      IF n_cursor%FOUND THEN
        -- parameterized cursor
        UPDATE N_Roll
        SET Name = o_record.Name, DOB = o_record.DOB, Phone = o_record.Phone,
        WHERE Rollno = n_cursor.Rollno;
      ELSE
        INSERT INTO N_Roll VALUES (o_record.Rollno, o_record.Name, o_record.D
      END IF;
    END LOOP;

    CLOSE n_cursor;
  END LOOP;
END;
/
```

Q6)a Library database with the schema Books(AccNo,Title,Author,Publisher,Count).
a.Create a table Library_Audit with same fields as of Books.
b.Create a before trigger to insert records into Librry_Audit table if there is deletion in Books table.
c.Create a after trigger to insert records into Librry_Audit table if there is updating in Books table.

Create

```
mysql> show tables;
+-------------------------+
| Tables_in_assignment_5_f |
+-------------------------+
| book                    |
+-------------------------+
1 row in set (0.00 sec)
```

Table: Book

```
mysql> desc book;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Acc_no    | varchar(10) | NO   | PRI | NULL    |       |
| Tittle    | varchar(30) | YES  |     | NULL    |       |
| Author    | varchar(30) | YES  |     | NULL    |       |
| Publisher | varchar(40) | YES  |     | NULL    |       |
| Count     | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Record: Book.

```
mysql> select * from book;
+--------+-----------------------------+--------------------+-----------------------+-------+
| Acc_no | Tittle                      | Author             | Publisher             | Count |
+--------+-----------------------------+--------------------+-----------------------+-------+
| B001   | To Kill a Mockingbird       | Harper Lee         | J. B. Lippincott      |     3 |
| B002   | 1984                        | George Orwell      | Secker and Warburg    |     5 |
| B003   | The Catcher in the Rye      | J.D.Salinger       | Little Brown and Co.  |     2 |
| B004   | The Great Garsby            | F.Scoot Fitzgerland | Little, Brown and Co. |     2 |
| B005   | One Hundred Years of Solitude | Gabriel Marquez  | Harper                |     1 |
+--------+-----------------------------+--------------------+-----------------------+-------+
5 rows in set (0.00 sec)
```

Queries:

1) Create a table Library_Audit with same fiels as of Books.

```
CREATE TRIGGER audit_after_update
AFTER UPDATE ON Books
FOR EACH ROW
BEGIN
  INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count)
  VALUES (NEW.AccNo, NEW.Title, NEW.Author, NEW.Publisher, NEW.Count);
END;
```

```
mysql> create table library_audit(
    -> Acc_no varchar(10) primary key,
    -> Title varchar(30)
    -> ,
    -> Author varchar(30),
    -> Publisher varchar(40),
    -> Count int)1
    -> ;
```

```
mysql> desc library_audit;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Acc_no    | varchar(10) | NO   | PRI | NULL    |       |
| Title     | varchar(30) | YES  |     | NULL    |       |
| Author    | varchar(30) | YES  |     | NULL    |       |
| Publisher | varchar(40) | YES  |     | NULL    |       |
| Count     | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

2) Create a before trigger to insert records into Librry_Audit table if there is deletion in Books table.

```
CREATE TRIGGER audit_before_delete
BEFORE DELETE ON Books
FOR EACH ROW
BEGIN
  INSERT INTO Library_Audit (AccNo, Title, Author, Publisher, Count)
  VALUES (OLD.AccNo, OLD.Title, OLD.Author, OLD.Publisher, OLD.Count);
END;
```

3) Create a after trigger to insert records into Librry_Audit table if there is updating in Books table.

Q7) a procedure called USER_QUERY_EMP that accepts three parameters. Parameter p_myeno is of IN parameter mode which provides the empno value. The other two parameters p_myjob and p_mysal are of OUT mode. The procedure retrieves the salary and job of an employee with the provided employee number and assigns those to the two OUT parameters respectively. The procedure should handle the error if the empno does not exist in the EMP table by displaying an appropriate message. Use bind variables for the two OUT Parameters. Compile the code, invoke the procedure, and display the salary and job title for employee number 7839. Do the same for employee number 7123.

Create

```
CREATE OR REPLACE PROCEDURE USER_QUERY_EMP (
    p_myeno IN EMP.EMPNO%TYPE,
    p_myjob OUT EMP.JOB%TYPE,
    p_mysal OUT EMP.SAL%TYPE
)
AS
BEGIN
    SELECT JOB, SAL
    INTO p_myjob, p_mysal
    FROM EMP
    WHERE EMPNO = p_myeno;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Employee number ' || p_myeno || ' does no
END;
/
```

```
SET SERVEROUTPUT ON;
DECLARE
    v_job EMP.JOB%TYPE;
    v_sal EMP.SAL%TYPE;
BEGIN
    USER_QUERY_EMP(p_myeno => 7839, p_myjob => v_job, p_mysal => v_sal);
    DBMS_OUTPUT.PUT_LINE('Employee number: 7839');
    DBMS_OUTPUT.PUT_LINE('Job: ' || v_job);
    DBMS_OUTPUT.PUT_LINE('Salary: ' || v_sal);

    USER_QUERY_EMP(p_myeno => 7123, p_myjob => v_job, p_mysal => v_sal);
    DBMS_OUTPUT.PUT_LINE('Employee number: 7123');
    DBMS_OUTPUT.PUT_LINE('Job: ' || v_job);
    DBMS_OUTPUT.PUT_LINE('Salary: ' || v_sal);
END;
/
```

```
Employee number: 7839
Job: PRESIDENT
Salary: 5000
Employee number 7123 does not exist.
```

Q8) Create a function named USER_ANNUAL_COMP that has three parameters p_eno, p_sal and p_comm for passing on the values of an employee number, the current salary and commission of the employee respectively. The function calculates and returns the annual compensation of the employee by using the following formula. annual_compensation = (p_sal+p_comm)*12 If the salary or commission value is NULL then zero should be substituted for it. Give a call to USER_ANNUAL_COMP from a SELECT statement, against the EMP table.

```
CREATE OR REPLACE FUNCTION USER_ANNUAL_COMP (
    p_eno IN EMP.EMPNO%TYPE,
    p_sal IN EMP.SAL%TYPE,
    p_comm IN EMP.COMM%TYPE
)
RETURN NUMBER
AS
    v_annual_comp NUMBER := 0;
BEGIN
    v_annual_comp := (COALESCE(p_sal, 0) + COALESCE(p_comm, 0)) * 12;
    RETURN v_annual_comp;
END;
/
```

```
SELECT EMPNO, ENAME, USER_ANNUAL_COMP(EMPNO, SAL, COMM) AS ANNUAL_COMP
FROM EMP;
```

Q9) Create a function named USER_VALID_DEPTNO that has a single parameter p_dno to accept a department number and returns a BOOLEAN value. The function returns TRUE if the department number exists in the DEPT table else it returns FALSE.

Here is the PL/SQL code for the function `USER_VALID_DEPTNO`:

sql

Copy code

```
CREATE OR REPLACE FUNCTION USER_VALID_DEPTNO (p_dno IN DEPT.DEPTNO%TYPE) RET
AS
    v_count NUMBER := 0;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM DEPT
    WHERE DEPTNO = p_dno;

    IF v_count = 1 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
/
```

```
DECLARE
    v_result BOOLEAN;
BEGIN
    v_result := USER_VALID_DEPTNO(10);
    IF v_result THEN
        DBMS_OUTPUT.PUT_LINE('Department number 10 exists.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Department number 10 does not exist.');
    END IF;

    v_result := USER_VALID_DEPTNO(50);
    IF v_result THEN
        DBMS_OUTPUT.PUT_LINE('Department number 50 exists.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Department number 50 does not exist.');
    END IF;
END;
/
```

```
Department number 10 exists.
Department number 50 does not exist.
```

Q10) Create a table named salaryLog with the appropriate columns and insert the empno, new grade, old salary and new salary values in salaryLog table if the grade of an employee changes. Create a trigger named TR_CHECK_GRADE that will be fired when a user modifies the EMP table. It will check whether the grade has changed by making use of the SALGRADE table. (Grade is dependent on Salary.) If grade is changed, the trigger will log the corresponding employee number, old salary, new salary and new grade into salaryLog table. Test the working of the trigger by firing an appropriate DML query

```
CREATE TABLE salaryLog (
    empno NUMBER,
    new_grade NUMBER,
    old_salary NUMBER,
    new_salary NUMBER,
    log_date DATE DEFAULT SYSDATE
);

CREATE OR REPLACE TRIGGER TR_CHECK_GRADE
AFTER UPDATE OF SAL, GRADE ON EMP
FOR EACH ROW
DECLARE
    v_old_grade NUMBER;
    v_new_grade NUMBER;
    v_old_salary NUMBER;
    v_new_salary NUMBER;
BEGIN
    -- Get the old and new grade values
    SELECT GRADE
    INTO v_old_grade
    FROM SALGRADE
    WHERE LOSAL <= :OLD.SAL AND HISAL >= :OLD.SAL;
```

```
    SELECT GRADE
    INTO v_new_grade
    FROM SALGRADE
    WHERE LOSAL <= :NEW.SAL AND HISAL >= :NEW.SAL;

    -- Check if grade has changed
    IF v_old_grade != v_new_grade THEN
        -- Log the changes in the salaryLog table
        v_old_salary := :OLD.SAL;
        v_new_salary := :NEW.SAL;

        INSERT INTO salaryLog (empno, new_grade, old_salary, new_salary)
        VALUES (:NEW.EMPNO, v_new_grade, v_old_salary, v_new_salary);
    END IF;
END;
/
```

```
UPDATE EMP SET SAL = 5000, GRADE = 5 WHERE EMPNO = 7369;
```

```
SELECT * FROM salaryLog;
```