

Practical No. 04

Name :Karan Mohite

TE- B 69

Aim : Write at least 10 SQL queries for suitable database application using SQL DML statements , Join , Sub-Query and View.

```
mysql> create database c4;
```

Query OK, 1 row affected (0.03 sec)

```
mysql> use c4;
```

Database changed

```
mysql> create table branch(branch_name varchar(20) primary key,branch_city  
varchar(20),assets int);
```

Query OK, 0 rows affected (0.04 sec)

```
mysql> create table customer(customer_name varchar(20) primary key,customer_street  
varchar(20), customer_city varchar(20));
```

```
mysql> create table account(account_number int primary key, branch_name varchar(20),  
amount int,foreign key (branch_name) references branch(branch_name));
```

```
mysql> create table loan(loan_number int primary key,branch_name varchar(20), amount  
int,foreign key (branch_name) references branch(branch_name));
```

```
mysql> create table depositor (customer_name varchar(20),account_number int,primary  
key(customer_name, account_number),foreign key (customer_name) references  
customer(customer_name),foreign key (account_number) references account(account_number));
```

```
mysql> create table borrower(customer_name varchar(20),loan_number int,foreign key
```

(customer_name) references customer(customer_name),foreign key (loan_number) references loan(loan_number));

```
mysql> insert into branch
values("akurdi","pune",12000),("loni","pune",67000),("wagholi","pune",5699),("latur","latur",4
5888);
```

Query OK, 4 rows affected (0.02 sec)

Records: 4 Duplicates: 0 Warnings: 0

```
mysql> select * from branch;
```

branch_name	branch_city	assets
akurdi	pune	12000
latur	latur	45888
loni	pune	67000
wagholi	pune	5699

4 rows in set (0.01 sec)

```
mysql> insert into customer
values("Karan","balajinagar","Lonavala"),("ishwari","shivajinagar","satara"),("Pranjali","lokma
nyatilk","pune"),("srushti","mg","latur");
```

Query OK, 4 rows affected (0.02 sec)

Records: 4 Duplicates: 0 Warnings: 0

```
mysql> select * from customer;
```

```
+-----+-----+-----+
| customer_name | customer_street | customer_city |
+-----+-----+-----+
| ishvari      | shivajinagar    | satara        |
| Pranjali     | lokmanyatilk    | pune          |
| Karan        | balajinagar     | Lonavala      |
| srushti      | mg              | latur         |
+-----+-----+-----+
```

4 rows in set (0.00 sec)

```
mysql> insert into account
```

```
values(101,"akurdi",12000),(102,"latur",200000),(103,"loni",40000),(104,"wagholi",23000);
```

Query OK, 4 rows affected (0.01 sec)

Records: 4 Duplicates: 0 Warnings: 0

```
mysql> select * from account;
```

```
+-----+-----+-----+
| account_number | branch_name | amount |
+-----+-----+-----+
| 101 | akurdi | 12000 |
| 102 | latur | 200000 |
| 103 | loni | 40000 |
| 104 | wagholi | 23000 |
```

```
+-----+-----+-----+
```

4 rows in set (0.00 sec)

```
mysql> insert into loan values(201,"akurdi",23000),(202,"latur",56000);
```

Query OK, 2 rows affected (0.02 sec)

Records: 2 Duplicates: 0 Warnings: 0

```
mysql> select * from loan;
```

```
+-----+-----+-----+
```

```
| loan_number | branch_name | amount |
```

```
+-----+-----+-----+
```

```
|      201 | akurdi      | 23000 |
```

```
|      202 | latur       | 56000 |
```

```
+-----+-----+-----+
```

2 rows in set (0.00 sec)

```
mysql> insert into depositor values("ishwari",101),("Pranjali",102);
```

Query OK, 2 rows affected (0.02 sec)

Records: 2 Duplicates: 0 Warnings: 0

```
mysql> select * from depositor;
```

```
+-----+-----+
```

```
| customer_name | account_number |
```

```
+-----+-----+
```

```
| ishwari      |      101 |
```

Pranjali		102	
----------	--	-----	--

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

2 rows in set (0.00 sec)

mysql> insert into borrower values("Karan",201),("srushti",202);

Query OK, 2 rows affected (0.01 sec)

Records: 2 Duplicates: 0 Warnings: 0

mysql> select * from borrower;

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

customer_name		loan_number	
---------------	--	-------------	--

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

Karan		201	
-------	--	-----	--

srushti		202	+-----+	+-----+
---------	--	-----	---------	---------

+

2 rows in set (0.00 sec)

1)Creating View on borrower table.

mysql> create view viewb as select customer_name, loan_number from borrower;

Query OK, 0 rows affected (0.01 sec)

mysql> select * from viewb;

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

customer_name		loan_number	
---------------	--	-------------	--

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

Karan		201	
-------	--	-----	--

srushti	202
---------	-----

```

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

```

2 rows in set (0.01 sec)

2) find the name of all branches in the loan relation

```
mysql> select branch_name from loan;
```

```

+-----+
| branch_name |
+-----+

```

```
| akurdi |
```

```
| latur |
```

```

+-----+

```

2 rows in set (0.01 sec)

3) find all loan number for loan where at Akurdi branch with loan amount < 12000

```
mysql> select loan_number,amount from Loan where branch_name='Akurdi' and
amount>12000;
```

```

+-----+-----+
| loan_number | amount |
+-----+-----+
| 201 | 23000 | +-----+-----+
+

```

1 row in set (0.01 sec)

4) Find all the customer who have a loan from bank & find there names, loan number , loan amount

```
mysql> select borrower.customer_name, loan.loan_number, loan.amount from borrower inner join loan on borrower.loan_number = loan.loan_number;
```

```
+-----+-----+-----+
| customer_name | loan_number | amount |
+-----+-----+-----+
| Karan        | 201        | 23000  |
| srushti      | 202        | 56000  |
+-----+-----+-----+
```

2 rows in set (0.00 sec)

5) Display the customer those who have loan account.

```
mysql> select * from loan inner join borrower on loan.loan_number = borrower.loan_number;
```

```
+-----+-----+-----+-----+-----+
| loan_number | branch_name | amount | customer_name | loan_number |
+-----+-----+-----+-----+-----+
| 201        | akurdi      | 23000  | Karan        | 201        |
| 202        | latur       | 56000  | srushti      | 202        |
+-----+-----+-----+-----+-----+
```

2 rows in set (0.00 sec)

6) List all customer in alphabetical order who have loan from Akurdi branch

```
mysql> select customer_name from borrower b join loan l on l.loan_number=b.loan_number order by customer_name;
```

```

+-----+
| customer_name |
+-----+
| renuka      |
| srushti     |
+-----+

```

2 rows in set (0.00 sec)

7) Find the average account balance at each branch

```
mysql> select branch_name ,avg(amount) from account group by
branch_name;
```

```

+-----+-----+
| branch_name | avg(amount) |
+-----+-----+
| akurdi     | 12000.0000 |
| latur      | 200000.0000 |
| loni       | 40000.0000 |
| wagholi    | 23000.0000 |
+-----+-----+

```

4 rows in set (0.02 sec)

8) Find no of depositors at each branch

```
mysql> select count(customer_name) from
depositor;
```

```

+-----+

```



```
| count(customer_name) |
```

```
+-----+
```

```
|          2 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

9) Calculate total loan amount given by bank

```
mysql> select sum(amount) from loan;
```

```
+-----+
```

```
| sum(amount) |
```

```
+-----+
```

```
|    79000 |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

10) Delete all loan with loan where amount between 13000 AND 15000

```
mysql> DELETE FROM loan WHERE amount BETWEEN 13000 AND 15000;
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from loan;
```

```
+-----+-----+-----+
```

```
| loan_number | branch_name | amount |
```

```
+-----+-----+-----+
```

```
|    201 | akurdi    | 23000 |
```

```
|    202 | latur     | 56000 |
```

Practical No. 2

create Database Assignment;

use Assignment;

create table Departments(dept_id INT PRIMARY KEY, dept_name VARCHAR(10));

create table Professors(

prof_id INT PRIMARY KEY, prof_fname VARCHAR(100), prof_lname VARCHAR(100),
designation VARCHAR(100), salary INT, doj DATE, email VARCHAR(255), phone INT, city
VARCHAR(100));

alter table Professors add dept_id INT;

alter table Professors add foreign key(dept_id) references Departments(dept_id) on delete cascade;

create table Shift(prof_id INT, shift VARCHAR(100), working_hours INT);

alter table Shift add foreign key(prof_id) references Professors(prof_id) on delete cascade;

create table works(dept_id INT, prof_id INT, duration YEAR, foreign key (dept_id) references
Departments on delete cascade, foreign key (prof_id) references Professors on delete cascade);

insert into Departments(dept_id,dept_name)

values (1, 'FE'), (2,'CE'),(3,'IT'),(4,'ENTC');

insert into Professors(prof_id,prof_fname,prof_lname,designation,salary,doj,email,phone,city,dept_id)
values (1,'Shivaji','Mundhe','Ast. Prof.',40000,'2009-07-01','svmundhe@pict.edu','24371101','Pune',1),

(2,'Nikhil','Sangade','Ast. Prof.',30000,'2019-03-
21','nvsangade@pict.edu','9762172013','Pune',1),

(3,'Kartik','Nandi','Prof.',50000,'1990-07-01','kcnandi@pict.edu','24371101','Pune',1),

(4,'Urmila','Pawar','Ast. Prof.',40000,'2016-01-
01','uspawar@pict.edu','7083664201','Mumbai',2),

(5,'Bhumesh','Masram','Ast. Prof.',30000,'2017-07-
15','bsmasram@pict.edu','24371101','Mumbai',2),

(6,'Archana','Ghotkar','Asc. Prof.',50000,'2000-03-
01','aaghotkar@pict.edu','24371101','Pune',2),

(7,'Girish','Mundada','Prof.',50000,'1900-09-01','gsmundada@pict.edu','24371101','Nashik',4),

(8,'Chetan','Pawar','Ast. Prof.',15000,'2018-02-
01','ccpawar@pict.edu','9028648563','Jalgaon',4),

```
(9,'Shweta','Dharmadhikari','Asc. Prof.',40000,'1995-12-31','scdharmadhikari@pict.edu','24371101','Pune',3),
```

```
(10,'Emmanuel','M','Prof.',50000,'2000-08-01','emmanuelm@pict.edu','24371101','Mumbai',3),
```

```
(11,'Mayuresh','Chavan','Ast. Prof',25000,'2015-01-01','mschavan@pict.edu','24567479','Sangli',2);
```

```
insert into Shift(shift,working_hours,prof_id)
```

```
values('morning',8,1),
```

```
      ('morning',8,2),
```

```
      ('afternoon',10,3),
```

```
      ('morning',8,4),
```

```
      ('afternoon',10,5),
```

```
      ('morning',8,6),
```

```
      ('morning',8,7),
```

```
      ('morning',8,8),
```

```
      ('afternoon',10,9),
```

```
      ('morning',8,10);
```

```
insert into works(dept_id,prof_id,duration)
```

```
values (1,1,9),
```

```
      (1,2,4),
```

```
      (1,3,27),
```

```
      (2,4,7),
```

```
      (2,5,3),
```

```
      (2,6,20),
```

```
      (4,7,29),
```

```
      (4,8,3),
```

```
      (3,9,16),
```

```
      (3,10,21);
```

/*2. Display all professors details with city pune and mumbai and professor first name starting with 'a' or 'd'.*/

```
select * from Professors where ((city='Pune' or city='Mumbai') and (prof_fname like 'A%' or prof_fname like 'D%'));
```

```
/*mysql> select * from Professors where ((city='Pune' or city='Mumbai') and (prof_fname like 'A%' or prof_fname like 'D%'));
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
```

```
| prof_id | prof_fname | prof_lname | designation | salary | doj      | email          | phone  | city |
dept_id |
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
```

```
|      6 | Archana   | Ghotkar    | Asc. Prof.  | 50000 | 2000-03-01 | aaghotkar@pict.edu | 24371101 |
Pune |      2 |
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
```

1 row in set (0.00 sec)

*/

/*3. list the number of different cities of professors.(use of distinct*/

```
select count(distinct city) from Professors;
```

/*

```
mysql> select count(distinct city) from Professors;
```

```
+-----+
```

```
| count(distinct city) |
```

```
+-----+
```

```
|          5 |
```

```
+-----+
```

1 row in set (0.00 sec)

*/

/*4. Give 5% increase in salary of the professors with date of joining 1-1-2015.*/

```
update Professors set salary=salary * 1.05 where date(doj) = '2015-01-01';
```

```
/*mysql> select * from Professors;
```

```

+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| prof_id | prof_fname | prof_lname | designation | salary | doj | email | phone |
city | dept_id |
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| 1 | Shivaji | Mundhe | Ast. Prof. | 40000 | 2009-07-01 | svmundhe@pict.edu |
24371101 | Pune | 1 |
| 2 | Nikhil | Sangade | Ast. Prof. | 30000 | 2019-03-21 | nvsangade@pict.edu |
9762172013 | Pune | 1 |
| 3 | Kartik | Nandi | Prof. | 50000 | 1990-07-01 | kcnandi@pict.edu | 24371101 |
Pune | 1 |
| 4 | Urmila | Pawar | Ast. Prof. | 40000 | 2016-01-01 | uspawar@pict.edu |
7083664201 | Mumbai | 2 |
| 5 | Bhumesh | Masram | Ast. Prof. | 30000 | 2017-07-15 | bsmasram@pict.edu |
24371101 | Mumbai | 2 |
| 6 | Archana | Ghotkar | Asc. Prof. | 50000 | 2000-03-01 | aaghotkar@pict.edu |
24371101 | Pune | 2 |
| 7 | Girish | Mundada | Prof. | 50000 | 1900-09-01 | gsmundada@pict.edu |
24371101 | Nashik | 4 |
| 8 | Chetan | Pawar | Ast. Prof. | 15000 | 2018-02-01 | ccpawar@pict.edu |
9028648563 | Jalgaon | 4 |
| 9 | Shweta | Dharmadhikari | Asc. Prof. | 40000 | 1995-12-31 | scdharmadhikari@pict.edu |
24371101 | Pune | 3 |
| 10 | Emmanuel | M | Prof. | 50000 | 2000-08-01 | emmanuelm@pict.edu |
24371101 | Mumbai | 3 |
| 11 | Mayuresh | Chavan | Ast. Prof | 26250 | 2015-01-01 | mschavan@pict.edu |
24567479 | Sangli | 2 |
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+

```

11 rows in set (0.00 sec)

*/

/*6. Find the names of professors belonging to pune or mumbai*/

select prof_fname from Professors where (city='Pune' or city='Mumbai');

/*

mysql> select prof_fname from Professors where (city='Pune' or city='Mumbai');

+-----+

| prof_fname |

```
+-----+
| Shivaji |
| Nikhil  |
| Kartik  |
| Urmila  |
| Bhumesh |
| Archana |
| Shweta  |
| Emmanuel |
+-----+
```

8 rows in set (0.00 sec)

*/

/*6. Find the names of professors belonging to pune or mumbai*/

select * from Professors where date(doj) = '2015-01-01' or date(doj) = '2016-01-01';

/*

mysql> select * from Professors where date(doj) = '2015-01-01' or date(doj) = '2016-01-01';

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+

| prof_id | prof_fname | prof_lname | designation | salary | doj       | email           | phone   | city |
dept_id |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+

| 4 | Urmila | Pawar | Ast. Prof. | 40000 | 2016-01-01 | uspawar@pict.edu | 7083664201 |
Mumbai | 2 |
| 11 | Mayuresh | Chavan | Ast. Prof | 26250 | 2015-01-01 | mschavan@pict.edu | 24567479 |
Sangli | 2 |
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
```

2 rows in set (0.00 sec)

*/

/*7. Find the professors who joined on date 1-1-2015 as well as in 1-1-2016*/

select * from Professors where salary = (select max(salary) from Professors);

/*

```
mysql> select * from Professors where salary = (select max(salary) from Professors);
```

prof_id	prof_fname	prof_lname	designation	salary	doj	email	phone	city	dept_id
3	Kartik	Nandi	Prof.	50000	1990-07-01	kcnandi@pict.edu	24371101	Pune	1
6	Archana	Ghotkar	Asc. Prof.	50000	2000-03-01	aaghotkar@pict.edu	24371101	Pune	2
7	Girish	Mundada	Prof.	50000	1900-09-01	gsmundada@pict.edu	24371101	Nashik	4
10	Emmanuel	M	Prof.	50000	2000-08-01	emmanuelm@pict.edu	24371101	Mumbai	3

4 rows in set (0.00 sec)

*/

/*8. Find the professor having maximum salary and names of professors having salary between 10,000 and 20,000.*/

```
select prof_fname, prof_lname from Professors where salary between 10000 and 20000;
```

/*

```
mysql> select prof_fname, prof_lname from Professors where salary between 10000 and 20000;
```

prof_fname	prof_lname
Chetan	Pawar

1 row in set (0.00 sec)

*/

/*9. Display all professors name with salary and date of joining with decreasing order of salary.*/

```
select prof_fname,prof_lname,salary,doj from Professors order by salary DESC;
```

```
/*
```

```
mysql> select prof_fname,prof_lname,salary,doj from Professors order by salary DESC;
```

```
+-----+-----+-----+-----+
| prof_fname | prof_lname | salary | doj      |
+-----+-----+-----+-----+
| Kartik    | Nandi      | 50000  | 1990-07-01 |
| Archana   | Ghotkar    | 50000  | 2000-03-01 |
| Girish    | Mundada    | 50000  | 1900-09-01 |
| Emmanuel  | M          | 50000  | 2000-08-01 |
| Shivaji   | Mundhe     | 40000  | 2009-07-01 |
| Urmila    | Pawar      | 40000  | 2016-01-01 |
| Shweta    | Dharmadhikari | 40000  | 1995-12-31 |
| Nikhil    | Sangade    | 30000  | 2019-03-21 |
| Bhumesh   | Masram     | 30000  | 2017-07-15 |
| Mayuresh  | Chavan     | 26250  | 2015-01-01 |
| Chetan    | Pawar      | 15000  | 2018-02-01 |
+-----+-----+-----+-----+
```

```
11 rows in set (0.00 sec)
```

```
*/
```

```
/*10. Display professors name, date of joining and dept_id with salary 30000, 40000 and 50000*/
```

```
create view salary_30k as select p.prof_fname,p.prof_lname,p.doj,p.dept_id from Professors as p
where salary = 30000;
```

```
/*
```

```
mysql> select * from salary_30k;
```

```
+-----+-----+-----+-----+
| prof_fname | prof_lname | doj      | dept_id |
+-----+-----+-----+-----+
| Nikhil    | Sangade    | 2019-03-21 | 1 |
| Bhumesh   | Masram     | 2017-07-15 | 2 |
+-----+-----+-----+-----+
```


Practical No. 3

```
mysql> use Assignment;
```

```
mysql> show tables;
```

```
+-----+
| Tables_in_Assignment1 |
+-----+
| Departments           |
| Professors            |
| Shift                 |
| salary_30k            |
| salary_40k            |
| salary_50k            |
| sequences_example     |
| works                 |
+-----+
```

```
mysql> select * from Professors;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| prof_id | prof_fname | prof_lname | designation | salary | doj       | email                | phone |
city    | dept_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| 1 | Shivaji | Mundhe | Ast. Prof. | 40000 | 2009-07-01 | svmundhe@pict.edu | 24371101 |
Pune | 1 |
| 2 | Nikhil | Sangade | Ast. Prof. | 30000 | 2019-03-21 | nvsangade@pict.edu | 9762172013 |
Pune | 1 |
| 3 | Kartik | Nandi | Prof. | 50000 | 1990-07-01 | kcnandi@pict.edu | 24371101 |
Pune | 1 |
| 4 | Urmila | Pawar | Ast. Prof. | 40000 | 2016-01-01 | uspawar@pict.edu | 7083664201 |
Mumbai | 2 |
| 5 | Bhumes | Masram | Ast. Prof. | 30000 | 2017-07-15 | bsmasram@pict.edu | 24371101 |
Mumbai | 2 |
| 6 | Archana | Ghotkar | Asc. Prof. | 50000 | 2000-03-01 | aaghotkar@pict.edu | 24371101 |
Pune | 2 |
| 7 | Girish | Mundada | Prof. | 50000 | 1900-09-01 | gsmundada@pict.edu | 24371101 |
Nashik | 4 |
```

8	Chetan	Pawar	Ast. Prof.	15000	2018-02-01	ccpawar@pict.edu
9028648563	Jalgaon	4				

9	Shweta	Dharmadhikari	Asc. Prof.	40000	1995-12-31	scdharmadhikari@pict.edu
24371101	Pune	3				

10	Emmanuel	M	Prof.	50000	2000-08-01	emmanuelm@pict.edu
24371101	Mumbai	3				

11	Mayuresh	Chavan	Ast. Prof	26250	2015-01-01	mschavan@pict.edu
24567479	Sangli	2				

```
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
```

```
mysql> select * from Departments;
```

```
+-----+-----+
| dept_id | dept_name |
+-----+-----+
| 1 | FE |
| 2 | CE |
| 3 | IT |
| 4 | ENTC |
| 5 | ME |
+-----+-----+
```

```
mysql> select * from Shift;
```

```
+-----+-----+-----+
| working_hours | prof_id | shift |
+-----+-----+-----+
| 8 | 1 | morning |
| 8 | 2 | morning |
| 10 | 3 | afternoon |
| 8 | 4 | morning |
| 10 | 5 | afternoon |
| 8 | 6 | morning |
| 8 | 7 | morning |
| 8 | 8 | morning |
| 10 | 9 | afternoon |
```

8	10	morning
10	11	afternoon

```
mysql> select * from works;
```

dept_id	prof_id	duration
1	1	9
1	2	4
1	3	27
2	4	7
2	5	3
2	6	20
4	7	29
4	8	3
3	9	16
3	10	21
2	11	6

#Queries:

/*1.Find the professor details and department details using NATURAL JOIN*/

```
mysql> select prof_id,prof_fname,prof_lname,dept_id,dept_name from Professors natural join Departments;
```

prof_id	prof_fname	prof_lname	dept_id	dept_name
1	Shivaji	Mundhe	1	FE
2	Nikhil	Sangade	1	FE
3	Kartik	Nandi	1	FE
4	Urmila	Pawar	2	CE
5	Bhumesh	Masram	2	CE

6	Archana	Ghotkar	2	CE
11	Mayuresh	Chavan	2	CE
9	Shweta	Dharmadhikari	3	IT
10	Emmanuel	M	3	IT
7	Girish	Mundada	4	ENTC
8	Chetan	Pawar	4	ENTC

```

/*2.Find the prof_id, prof_name and shift*/

```

```

select p.prof_id,p.prof_fname,p.prof_lname,s.shift from Professors as p inner join Shift as s on
s.prof_id = p.prof_id order by p.prof_id;

```

prof_id	prof_fname	prof_lname	shift
1	Shivaji	Mundhe	morning
2	Nikhil	Sangade	morning
3	Kartik	Nandi	afternoon
4	Urmila	Pawar	morning
5	Bhumesh	Masram	afternoon
6	Archana	Ghotkar	morning
7	Girish	Mundada	morning
8	Chetan	Pawar	morning
9	Shweta	Dharmadhikari	afternoon
10	Emmanuel	M	morning
11	Mayuresh	Chavan	afternoon

```

/*3.List all the department details and the corresponding names of professors in the same
department.*/

```

```

select d.dept_id,d.dept_name,p.prof_fname,p.prof_lname from Departments as d left join Professors
as p on p.dept_id = d.dept_id;

```

dept_id	dept_name	prof_fname	prof_lname
---------	-----------	------------	------------

1	FE	Shivaji	Mundhe	
1	FE	Nikhil	Sangade	
1	FE	Kartik	Nandi	
2	CE	Urmila	Pawar	
2	CE	Bhumesh	Masram	
2	CE	Archana	Ghotkar	
2	CE	Mayuresh	Chavan	
3	IT	Shweta	Dharmadhikari	
3	IT	Emmanuel	M	
4	ENTC	Girish	Mundada	
4	ENTC	Chetan	Pawar	
5	ME	NULL	NULL	

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

/*4.List all the professors and the corresponding names of department*/

select * from Professors as p inner join Departments as d on p.dept_id = d.dept_id;

prof_id	prof_fname	prof_lname	designation	salary	doj	email	phone
city	dept_id	dept_id	dept_name				

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

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

1	Shivaji	Mundhe	Ast. Prof.	40000	2009-07-01	svmundhe@pict.edu	
24371101	Pune	1	1 FE				
2	Nikhil	Sangade	Ast. Prof.	30000	2019-03-21	nvsangade@pict.edu	
9762172013	Pune	1	1 FE				
3	Kartik	Nandi	Prof.	50000	1990-07-01	kcnandi@pict.edu	24371101
Pune	1	1 FE					
4	Urmila	Pawar	Ast. Prof.	40000	2016-01-01	uspawar@pict.edu	
7083664201	Mumbai	2	2 CE				
5	Bhumesh	Masram	Ast. Prof.	30000	2017-07-15	bsmasram@pict.edu	
24371101	Mumbai	2	2 CE				
6	Archana	Ghotkar	Asc. Prof.	50000	2000-03-01	aaghotkar@pict.edu	
24371101	Pune	2	2 CE				
11	Mayuresh	Chavan	Ast. Prof	26250	2015-01-01	mschavan@pict.edu	
24567479	Sangli	2	2 CE				
9	Shweta	Dharmadhikari	Asc. Prof.	40000	1995-12-31	scdharmadhikari@pict.edu	
24371101	Pune	3	3 IT				

10	Emmanuel	M	Prof.	50000	2000-08-01	emmanuelm@pict.edu
24371101	Mumbai	3	3	IT		

7	Girish	Mundada	Prof.	50000	1900-09-01	gsmundada@pict.edu
24371101	Nashik	4	4	ENTC		

8	Chetan	Pawar	Ast. Prof.	15000	2018-02-01	ccpawar@pict.edu
9028648563	Jalgaon	4	4	ENTC		

```

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

```

```
/*5.Display professor name, dept_name, shift, salary where prof_id = 11;*/
```

```
select p.prof_fname,p.prof_lname,d.dept_name,s.shift,p.salary from Professors as p inner join
Departments as d on d.dept_id = p.dept_id inner join Shift as s on s.prof_id = p.prof_id where
p.prof_id=11;
```

```

+-----+-----+-----+-----+-----+
| prof_fname | prof_lname | dept_name | shift | salary |
+-----+-----+-----+-----+-----+
| Mayuresh | Chavan | CE | afternoon | 26250 |
+-----+-----+-----+-----+-----+

```

```
/*6.list the total number of professor in each department.*/
```

```
select count(p.prof_id),d.dept_name from Professors as p right join Departments as d on d.dept_id =
p.dept_id group by d.dept_id;
```

```

+-----+-----+
| count(p.prof_id) | dept_name |
+-----+-----+
| 3 | FE |
| 4 | CE |
| 2 | IT |
| 2 | ENTC |
| 0 | ME |
+-----+-----+

```

```
/*7. List the prof_id associated department and the dept_name having name 'ce';*/
```

```
select p.prof_id,p.dept_id,d.dept_name from Professors as p inner join Departments as d on p.dept_id
= d.dept_id having d.dept_name="CE";
```

```

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

```

prof_id	dept_id	dept_name
4	2	CE
5	2	CE
6	2	CE
11	2	CE

/*8. Find the names of all departments where the professors joined in year 2015 (or date of joining is 1-1-2015).*/

```
select d.dept_name,p.prof_id,p.prof_fname,p.prof_lname from Professors as p inner join Departments as d on p.dept_id = d.dept_id where p.doj="2015-01-01";
```

dept_name	prof_id	prof_fname	prof_lname
CE	11	Mayuresh	Chavan

/*9.Create view showing the professor and shift details*/

```
create view profShift as select p.prof_id,p.prof_fname,p.prof_lname,s.shift from Professors as p inner join Shift as s on s.prof_id = p.prof_id order by p.prof_id;
```

```
mysql> select * from profShift;
```

prof_id	prof_fname	prof_lname	shift
1	Shivaji	Mundhe	morning
2	Nikhil	Sangade	morning
3	Kartik	Nandi	afternoon
4	Urmila	Pawar	morning
5	Bhumesh	Masram	afternoon
6	Archana	Ghotkar	morning
7	Girish	Mundada	morning
8	Chetan	Pawar	morning
9	Shweta	Dharmadhikari	afternoon
10	Emmanuel	M	morning

Code:

```
SET SERVEROUT ON
SET VERIFY OFF
/*
CREATE TABLE borrower(roll_no NUMBER , name VARCHAR2(25), dateofissue
DATE,name_of_book VARCHAR2(25), status VARCHAR2(20));
CREATE TABLE fine(roll_no NUMBER,date_of_return DATE,amt NUMBER);
INSERT INTO borrower VALUES(45,'ASHUTOSH',TO_DATE('01-08-2022','DD-MM-
YYYY'),'HARRY POTTER','PENDING');
INSERT INTO borrower VALUES(46,'ARYAN',TO_DATE('15-08-2022','DD-MM-
YYYY'),'DARK MATTER','PENDING');
INSERT INTO borrower VALUES(47,'ROHAN',TO_DATE('24-08-2022','DD-MM-
YYYY'),'SILENT HILL','PENDING');
INSERT INTO borrower VALUES(48,'SANKET',TO_DATE('26-08-2022','DD-MM-
YYYY'),'GOD OF WAR','PENDING');
INSERT INTO borrower VALUES(49,'SARTHAK',TO_DATE('09-09-2022','DD-MM-
YYYY'),'SPIDER-MAN','PENDING');
*/
DECLARE
    i_roll_no NUMBER;
    name_of_book VARCHAR2(25);
    no_of_days NUMBER;
    return_date DATE := TO_DATE(SYSDATE,'DD-MM-YYYY');
    temp NUMBER;
    doi DATE;
    fine NUMBER;
BEGIN
    i_roll_no := &i_roll_no;
    name_of_book := '&nameofbook';
    --dbms_output.put_line(return_date);
    SELECT to_date(borrower.dateofissue,'DD-MM-YYYY') INTO doi FROM
borrower WHERE borrower.roll_no = i_roll_no AND borrower.name_of_book =
name_of_book;
    no_of_days := return_date-doi;
    dbms_output.put_line(no_of_days);
    IF (no_of_days >15 AND no_of_days <=30) THEN
        fine := 5*no_of_days;

    ELSIF (no_of_days>30 ) THEN
        temp := no_of_days-30;
        fine := 150 + temp*50;
    END IF;
    dbms_output.put_line(fine);
    INSERT INTO fine VALUES(i_roll_no,return_date,fine);
    UPDATE borrower SET status = 'RETURNED' WHERE borrower.roll_no =
i_roll_no;

END;
/
/*
-----INPUT-----

Enter value for i_roll_no: 46
Enter value for nameofbook: DARK MATTER
```


-----OUTPUT-----

BORROWER:

ROLL_NO	NAME	DATEOFISSUE	NAME_OF_BOOK	STATUS
45	ASHUTOSH	01-08-22	HARRY POTTER	PENDING
46	ARYAN	15-08-22	DARK MATTER	RETURNED
47	ROHAN	24-08-22	SILENT HILL	PENDING
48	SANKET	26-08-22	GOD OF WAR	PENDING
49	SARTHAK	09-09-22.	SPIDER-MAN	PENDING

FINE:

ROLL_NO	DATEOFRE	AMOUNT
46	14-09-22	150

.....INPUT.....

Enter value for i_roll_no: 45
Enter value for nameofbook: HARRY POTTER

-----OUTPUT-----

BORROWER:

ROLL_NO.	NAME	DATEOFIS	NAME_OF_BOOK	STATUS
45	ASHUTOSH	01-08-22	HARRY POTTER	RETURNED
46	ARYAN	15-08-22	DARK MATTER	RETURNED
47	ROHAN	24-08-22	SILENT HILL	PENDING
48	SANKET	26-08-22	GOD OF WAR	PENDING
49.	SARTHAK	09-09-22	SPIDER-MAN	PENDING

FINE:

ROLL_NO	DATEOFRE	AMOUNT
46.	14-09-22	150
45	14-09-22	850

TITLE : Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function. Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is ≤ 1500 and marks ≥ 990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQL block to use procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll, Name, Class) Note: Instructor will frame the problem statement for writing stored procedure and Function in line with above statement.

```
CREATE DATABASE stud;
USE stud;
```

```
CREATE TABLE stud_marks (
    roll_no INTEGER PRIMARY KEY,
    name VARCHAR(20),
    total_marks INTEGER
);
```

```
CREATE TABLE result (
    roll_no INTEGER,
    class VARCHAR(20),
    CONSTRAINT xyz FOREIGN KEY (roll_no) REFERENCES stud_marks(roll_no)
);
```

```
INSERT INTO stud_marks (roll_no, name, total_marks) VALUES
(1, 'Mitali', 920),
(2, 'Divya', 1150),
(3, 'Rushali', 950),
(4, 'Poojs', 840),
(5, 'Rutuja', 1000),
(6, 'Rani', 860);
```

```
DELIMITER //
CREATE PROCEDURE proc_Grade(IN roll INTEGER)
BEGIN
    DECLARE m INTEGER;
    DECLARE c VARCHAR(20);
    SELECT total_marks INTO m FROM stud_marks WHERE roll_no = roll;

    IF m >= 990 AND m <= 1500 THEN
        SET c = 'Distinction';
    ELSEIF m >= 900 AND m <= 989 THEN
        SET c = 'First Class';
    ELSEIF m >= 825 AND m <= 899 THEN
        SET c = 'Higher Second Class';
    END IF;

    INSERT INTO result (roll_no, class) VALUES (roll, c);
END //
DELIMITER ;
```

```
CALL proc_Grade(1);
CALL proc_Grade(2);
CALL proc_Grade(3);
CALL proc_Grade(4);
CALL proc_Grade(5);
CALL proc_Grade(6);
```

```
SELECT * FROM result;
```

```
DELIMITER //
```

```
CREATE FUNCTION disp_grade2(roll_no INTEGER) RETURNS VARCHAR(20)
```

```
BEGIN
```

```
    DECLARE m INTEGER;
```

```
    DECLARE c VARCHAR(20);
```

```
    SELECT total_marks INTO m FROM stud_marks WHERE roll_no = roll_no;
```

```
    IF m >= 990 AND m <= 1500 THEN
```

```
        SET c = 'Distinction';
```

```
    ELSEIF m >= 900 AND m <= 989 THEN
```

```
        SET c = 'First Class';
```

```
    ELSEIF m >= 825 AND m <= 899 THEN
```

```
        SET c = 'Higher Second Class';
```

```
    END IF;
```

```
    RETURN c;
```

```
END //
```

```
DELIMITER ;
```

```
SELECT disp_grade2(1);
```

```
SELECT disp_grade2(2);
```

```
SELECT disp_grade2(3);
```

```
SELECT disp_grade2(4);
```

```
SELECT disp_grade2(5);
```

```
SELECT disp_grade2(6);
```

OUTPUT:

```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 8.0.39 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE stud;
Query OK, 1 row affected (0.01 sec)

mysql> USE stud;
Database changed
mysql> CREATE TABLE stud_marks (
  ->     roll_no INT PRIMARY KEY,
  ->     name VARCHAR(20),
  ->     total_marks INT
  -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE result (
  ->     roll_no INT,
  ->     class VARCHAR(20),
  ->     FOREIGN KEY (roll_no) REFERENCES stud_marks(roll_no)
  -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO stud_marks (roll_no, name, total_marks) VALUES
  -> (1, 'Mitali', 920),
  -> (2, 'Divya', 1150),
  -> (3, 'Rushali', 950),
  -> (4, 'Pooja', 840),
  -> (5, 'Rutuja', 1000),
  -> (6, 'Rani', 860);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> DELIMITER //
mysql> CREATE PROCEDURE proc_Grade(IN roll INT)
  -> BEGIN
  ->     DECLARE marks INT;
  ->     DECLARE category VARCHAR(20);
  ->
  ->     SELECT total_marks INTO marks FROM stud_marks WHERE roll_no = roll;
  ->
  ->     IF marks >= 990 AND marks <= 1500 THEN
  ->         SET category = 'Distinction';
  ->     ELSEIF marks >= 900 AND marks < 990 THEN
  ->         SET category = 'First Class';
```

Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> CALL proc_Grade(1);

Query OK, 1 row affected (0.01 sec)

mysql> CALL proc_Grade(2);

Query OK, 1 row affected (0.00 sec)

mysql> CALL proc_Grade(3);

Query OK, 1 row affected (0.00 sec)

mysql> CALL proc_Grade(4);

Query OK, 1 row affected (0.00 sec)

mysql> CALL proc_Grade(5);

Query OK, 1 row affected (0.00 sec)

mysql> CALL proc_Grade(6);

Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM result;

roll_no	class
1	First Class
2	Distinction
3	First Class
4	Higher Second Class
5	Distinction
6	Higher Second Class

6 rows in set (0.00 sec)

mysql> DELIMITER //

mysql> CREATE FUNCTION disp_grade(roll INT) RETURNS VARCHAR(20)

-> BEGIN

-> DECLARE marks INT;

-> DECLARE category VARCHAR(20);

->

-> SELECT total_marks INTO marks FROM stud_marks WHERE roll_no = roll;

->

-> IF marks >= 990 AND marks <= 1500 THEN

-> SET category = 'Distinction';

-> ELSEIF marks >= 900 AND marks < 990 THEN

-> SET category = 'First Class';

-> ELSEIF marks >= 825 AND marks < 900 THEN

-> SET category = 'Higher Second Class';

-> ELSE

-> SET category = 'Not Classified';

-> END IF;

->

-> RETURN category;

-> END //

Query OK, 0 rows affected (0.01 sec)

```
| disp_grade(1) |
+-----+
| First Class  |
+-----+
1 row in set (0.00 sec)

mysql> SELECT disp_grade(2);
+-----+
| disp_grade(2) |
+-----+
| Distinction  |
+-----+
1 row in set (0.00 sec)

mysql> SELECT disp_grade(3);
+-----+
| disp_grade(3) |
+-----+
| First Class  |
+-----+
1 row in set (0.00 sec)

mysql> SELECT disp_grade(4);
+-----+
| disp_grade(4) |
+-----+
| Higher Second Class |
+-----+
1 row in set (0.00 sec)

mysql> SELECT disp_grade(5);
+-----+
| disp_grade(5) |
+-----+
| Distinction  |
+-----+
1 row in set (0.00 sec)

mysql> SELECT disp_grade(6);
+-----+
| disp_grade(6) |
+-----+
| Higher Second Class |
+-----+
1 row in set (0.00 sec)

mysql> |
```

Assignment-6

Code:

```
-- Create the database and use it
CREATE DATABASE assi7;
USE assi7;

-- Create the old_roll and new_roll tables
CREATE TABLE old_roll (roll INT, name VARCHAR(10));
CREATE TABLE new_roll (roll INT, name VARCHAR(10));

-- Insert data into the old_roll table
INSERT INTO old_roll VALUES (4, 'D');
INSERT INTO old_roll VALUES (3, 'BCD');
INSERT INTO old_roll VALUES (1, 'BC');
INSERT INTO old_roll VALUES (5, 'BCH');

-- Insert data into the new_roll table with new names
INSERT INTO new_roll VALUES (2, 'Yash');
INSERT INTO new_roll VALUES (5, 'Varad');    -- Duplicate, will be skipped
INSERT INTO new_roll VALUES (1, 'Gauri');    -- Duplicate, will be skipped
INSERT INTO new_roll VALUES (6, 'Shravani');
INSERT INTO new_roll VALUES (7, 'Vishwajeet');

-- Display the initial data
SELECT * FROM old_roll;
SELECT * FROM new_roll;

-- Define the procedure to merge data
DELIMITER $$

CREATE PROCEDURE roll_list()
BEGIN
    DECLARE oldrollnumber INT;
    DECLARE oldname VARCHAR(10);
    DECLARE newrollnumber INT;
    DECLARE newname VARCHAR(10);
    DECLARE done INT DEFAULT FALSE;

    -- Declare cursors for old_roll and new_roll
    DECLARE c1 CURSOR FOR SELECT roll, name FROM old_roll;
    DECLARE c2 CURSOR FOR SELECT roll, name FROM new_roll;

    -- Continue handler for cursor not found
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the new_roll cursor
    OPEN c2;
```

```

loop1: LOOP
    -- Fetch new roll data
    FETCH c2 INTO newrollnumber, newname;
    IF done THEN
        LEAVE loop1;
    END IF;

    SET done = FALSE; -- Reset done for the old_roll cursor

    -- Open the old_roll cursor
    OPEN c1;

    loop2: LOOP
        -- Fetch old roll data
        FETCH c1 INTO oldrollnumber, oldname;
        IF done THEN
            LEAVE loop2;
        END IF;

        -- Check for duplicates
        IF oldrollnumber = newrollnumber THEN
            LEAVE loop2; -- Skip if duplicate found
        END IF;
    END LOOP;

    -- Insert if no duplicates were found
    IF done THEN
        INSERT INTO old_roll (roll, name) VALUES (newrollnumber, newname);
    END IF;

    CLOSE c1; -- Close old_roll cursor after processing
END LOOP;

CLOSE c2; -- Close new_roll cursor after processing
END $$

DELIMITER ;

-- Call the procedure to merge data
CALL roll_list();

-- Display the merged old_roll table
SELECT * FROM old_roll;

```


Output:

```
MySQL 8.0 Command Line Cli × +
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 67
Server version: 8.0.39 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> -- Create the database and use it
mysql> CREATE DATABASE assi7;
Query OK, 1 row affected (0.01 sec)

mysql> USE assi7;
Database changed
mysql>
mysql> -- Create the old_roll and new_roll tables
mysql> CREATE TABLE old_roll (roll INT, name VARCHAR(10));
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE new_roll (roll INT, name VARCHAR(10));
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> -- Insert data into the old_roll table
mysql> INSERT INTO old_roll VALUES (4, 'D');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO old_roll VALUES (3, 'BCD');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO old_roll VALUES (1, 'BC');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO old_roll VALUES (5, 'BCH');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> -- Insert data into the new_roll table with new names
mysql> INSERT INTO new_roll VALUES (2, 'Yash');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO new_roll VALUES (5, 'Varad');      -- Duplicate, will be skipped
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO new_roll VALUES (1, 'Gauri');      -- Duplicate, will be skipped
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO new_roll VALUES (6, 'Shravani');
Query OK, 1 row affected (0.00 sec)
```



```
mysql> INSERT INTO new_roll VALUES (7, 'Vishwajeet');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql>  
mysql> -- Display the initial data  
mysql> SELECT * FROM old_roll;
```

roll	name
4	D
3	BCD
1	BC
5	BCH

```
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM new_roll;
```

roll	name
2	Yash
5	Varad
1	Gauri
6	Shravani
7	Vishwajeet

```
5 rows in set (0.00 sec)
```

```
mysql>  
mysql> -- Define the procedure to merge data  
mysql> DELIMITER $$  
mysql>  
mysql> CREATE PROCEDURE roll_list()  
-> BEGIN  
->     DECLARE oldrollnumber INT;  
->     DECLARE oldname VARCHAR(10);  
->     DECLARE newrollnumber INT;  
->     DECLARE newname VARCHAR(10);  
->     DECLARE done INT DEFAULT FALSE;  
->  
->     -- Declare cursors for old_roll and new_roll  
->     DECLARE c1 CURSOR FOR SELECT roll, name FROM old_roll;  
->     DECLARE c2 CURSOR FOR SELECT roll, name FROM new_roll;  
->  
->     -- Continue handler for cursor not found  
->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  
->  
->     -- Open the new_roll cursor  
->     OPEN c2;  
->  
->     loop1: LOOP  
->         -- Fetch new roll data  
->         FETCH c2 INTO newrollnumber, newname;  
->         IF done THEN
```

```
->      IF done THEN
->          LEAVE loop1;
->      END IF;
->
->      SET done = FALSE; -- Reset done for the old_roll cursor
->
->      -- Open the old_roll cursor
->      OPEN c1;
->
->      loop2: LOOP
->          -- Fetch old roll data
->          FETCH c1 INTO oldrollnumber, oldname;
->          IF done THEN
->              LEAVE loop2;
->          END IF;
->
->          -- Check for duplicates
->          IF oldrollnumber = newrollnumber THEN
->              LEAVE loop2; -- Skip if duplicate found
->          END IF;
->      END LOOP;
->
->      -- Insert if no duplicates were found
->      IF done THEN
->          INSERT INTO old_roll (roll, name) VALUES (newrollnumber, newname);
->      END IF;
->
->      CLOSE c1; -- Close old_roll cursor after processing
->  END LOOP;
->
->      CLOSE c2; -- Close new_roll cursor after processing
->  END $$
```

Query OK, 0 rows affected (0.00 sec)

```
mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Call the procedure to merge data
mysql> CALL roll_list();
Query OK, 1 row affected (0.00 sec)
```

```
mysql>
mysql> -- Display the merged old_roll table
mysql> SELECT * FROM old_roll;
```

roll	name
4	D
3	BCD
1	BC
5	BCH
2	Yash

5 rows in set (0.00 sec)

Assignment-7

Code:

```
-- Step 1: Create the Library Table
CREATE TABLE Library (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    author VARCHAR(50),
    year_published INT
);

-- Step 2: Create the Library_Audit Table
CREATE TABLE Library_Audit (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    id INT,
    name VARCHAR(50),
    author VARCHAR(50),
    year_published INT,
    action VARCHAR(10),
    change_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Step 3: Create Row-Level Triggers

-- Trigger to log old values before an update
DELIMITER $$

CREATE TRIGGER before_update_library
BEFORE UPDATE ON Library
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (id, name, author, year_published, action)
    VALUES (OLD.id, OLD.name, OLD.author, OLD.year_published, 'UPDATE');
END$$

DELIMITER ;

-- Trigger to log old values before a delete
DELIMITER $$

CREATE TRIGGER before_delete_library
BEFORE DELETE ON Library
FOR EACH ROW
BEGIN
    INSERT INTO Library_Audit (id, name, author, year_published, action)
    VALUES (OLD.id, OLD.name, OLD.author, OLD.year_published, 'DELETE');
END$$
```

DELIMITER ;

-- Step 4: Create Statement-Level Triggers (Optional)
-- (Typically not necessary for auditing)

-- Trigger to log after an update (optional)
DELIMITER \$\$

```
CREATE TRIGGER after_update_library
AFTER UPDATE ON Library
BEGIN
    -- This could log additional information if desired
END$$
```

DELIMITER ;

-- Trigger to log after a delete (optional)
DELIMITER \$\$

```
CREATE TRIGGER after_delete_library
AFTER DELETE ON Library
BEGIN
    -- This could log additional information if desired
END$$
```

DELIMITER ;

-- Step 5: Test the Triggers

-- Insert example records into Library
INSERT INTO Library VALUES (1, 'Book A', 'Author A', 2020);
INSERT INTO Library VALUES (2, 'Book B', 'Author B', 2021);

-- Update a record
UPDATE Library SET name = 'Updated Book A' WHERE id = 1;

-- Delete a record
DELETE FROM Library WHERE id = 2;

-- Step 6: Check the Audit Log
SELECT * FROM Library_Audit;

-- Display current state of the Library table
SELECT * FROM Library;

Output:

```
MySQL 8.0 Command Line Cli  X  +  v

Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 70
Server version: 8.0.39 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database library;
Query OK, 1 row affected (0.01 sec)

mysql> use library;
Database changed
mysql> -- Step 1: Create the Library Table
mysql> CREATE TABLE Library (
  ->   id INT PRIMARY KEY,
  ->   name VARCHAR(50),
  ->   author VARCHAR(50),
  ->   year_published INT
  -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- Step 2: Create the Library_Audit Table
mysql> CREATE TABLE Library_Audit (
  ->   audit_id INT AUTO_INCREMENT PRIMARY KEY,
  ->   id INT,
  ->   name VARCHAR(50),
  ->   author VARCHAR(50),
  ->   year_published INT,
  ->   action VARCHAR(10),
  ->   change_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
  -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> -- Step 3: Create Row-Level Triggers
mysql>
mysql> -- Trigger to log old values before an update
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_update_library
  -> BEFORE UPDATE ON Library
  -> FOR EACH ROW
  -> BEGIN
  ->   INSERT INTO Library_Audit (id, name, author, year_published, action)
  ->   VALUES (OLD.id, OLD.name, OLD.author, OLD.year_published, 'UPDATE');
  -> END$$
Query OK, 0 rows affected (0.01 sec)
```

```
    -> VALUES (OLD.id, OLD.name, OLD.author, OLD.year_published, 'UPDATE');
    -> END$$
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Trigger to log old values before a delete
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER before_delete_library
    -> BEFORE DELETE ON Library
    -> FOR EACH ROW
    -> BEGIN
    ->     INSERT INTO Library_Audit (id, name, author, year_published, action)
    ->     VALUES (OLD.id, OLD.name, OLD.author, OLD.year_published, 'DELETE');
    -> END$$
Query OK, 0 rows affected (0.01 sec)
```

```
mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Step 4: Create Statement-Level Triggers (Optional)
mysql> -- (Typically not necessary for auditing)
mysql>
mysql> -- Trigger to log after an update (optional)
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER after_update_library
    -> AFTER UPDATE ON Library
    -> BEGIN
    ->     -- This could log additional information if desired
    -> END$$
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version [10.6.4] at line 3
```

```
mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Trigger to log after a delete (optional)
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER after_delete_library
    -> AFTER DELETE ON Library
    -> BEGIN
    ->     -- This could log additional information if desired
    -> END$$
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version [10.6.4] at line 3
```

```
mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Step 5: Test the Triggers
mysql>
```

END' at line 3

mysql>

mysql> DELIMITER ;

mysql>

mysql> -- Step 5: Test the Triggers

mysql>

mysql> -- Insert example records into Library

mysql> INSERT INTO Library VALUES (1, 'Book A', 'Author A', 2020);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Library VALUES (2, 'Book B', 'Author B', 2021);
Query OK, 1 row affected (0.00 sec)

mysql>

mysql> -- Update a record

mysql> UPDATE Library SET name = 'Updated Book A' WHERE id = 1;
Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql>

mysql> -- Delete a record

mysql> DELETE FROM Library WHERE id = 2;
Query OK, 1 row affected (0.00 sec)

mysql>

mysql> -- Step 6: Check the Audit Log

mysql> SELECT * FROM Library_Audit;

audit_id	id	name	author	year_published	action	change_time
1	1	Book A	Author A	2020	UPDATE	2024-10-20 12:12:14
2	2	Book B	Author B	2021	DELETE	2024-10-20 12:12:14

2 rows in set (0.00 sec)

mysql>

mysql> -- Display current state of the Library table

mysql> SELECT * FROM Library;

id	name	author	year_published
1	Updated Book A	Author A	2020

1 row in set (0.00 sec)

mysql>

Assignment-8

Code:

```
from flask import Flask, request, redirect, url_for, render_template
import mysql.connector

app = Flask(__name__)

# Database connection
def get_db_connection():
    return mysql.connector.connect(
        host='localhost',
        user='root',
        password='8011',
        database='testdb'
    )

@app.route('/')
def index():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM users')
    users = cursor.fetchall()
    cursor.close()
    conn.close()
    return render_template('index.html', users=users)

@app.route('/add', methods=['POST'])
def add_user():
    name = request.form['name']
    email = request.form['email']
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute('INSERT INTO users (name, email) VALUES (%s, %s)', (name, email))
    conn.commit()
    cursor.close()
    conn.close()
    return redirect(url_for('index'))

@app.route('/edit/<int:id>', methods=['GET', 'POST'])
def edit_user(id):
    conn = get_db_connection()
    cursor = conn.cursor()

    if request.method == 'POST':
```

```
name = request.form['name']
email = request.form['email']
cursor.execute('UPDATE users SET name = %s, email = %s WHERE id = %s', (name, email, id))
conn.commit()
cursor.close()
conn.close()
return redirect(url_for('index'))
```

```
cursor.execute('SELECT * FROM users WHERE id = %s', (id,))
user = cursor.fetchone()
cursor.close()
conn.close()
return render_template('edit.html', user=user)
```

```
@app.route('/delete/<int:id>', methods=['POST'])
def delete_user(id):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute('DELETE FROM users WHERE id = %s', (id,))
    conn.commit()
    cursor.close()
    conn.close()
    return redirect(url_for('index'))
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Output:



User Management

- rohan yeole - rohanyeole@gmail.com [Edit](#)
- yash kamathe - yashkamathe@gmail.com [Edit](#)
- om shete - omshete@gmail.com [Edit](#)

Activate Windows
Go to Settings to activate Windows.

```
Welcome | app.py | index.html | edit.html | python x
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility
purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\Admin\Desktop\dmsl 8> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inste
ad.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 685-161-179
127.0.0.1 - - [21/Oct/2024 13:15:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Oct/2024 13:15:59] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [21/Oct/2024 13:16:04] "POST /delete/1 HTTP/1.1" 302 -
127.0.0.1 - - [21/Oct/2024 13:16:04] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Oct/2024 13:16:04] "POST /delete/2 HTTP/1.1" 302 -
127.0.0.1 - - [21/Oct/2024 13:16:04] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Oct/2024 13:16:05] "POST /delete/3 HTTP/1.1" 302 -
127.0.0.1 - - [21/Oct/2024 13:16:05] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Oct/2024 13:16:35] "POST /add HTTP/1.1" 302 -
127.0.0.1 - - [21/Oct/2024 13:16:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Oct/2024 13:16:46] "POST /add HTTP/1.1" 302 -
127.0.0.1 - - [21/Oct/2024 13:16:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Oct/2024 13:17:02] "POST /add HTTP/1.1" 302 -
127.0.0.1 - - [21/Oct/2024 13:17:02] "GET / HTTP/1.1" 200 -

Activate Windows
Go to Settings to activate Windows.
```


Assignment-9

Code:

```
// Switch to the LibraryDB database
use LibraryDB;
// Create (Insert) - Add two books to the collection
db.library.insertMany([
  { "bid": 1, "name": "C++", "author": "Bjarne Stroustrup", "cost": 300 },
  { "bid": 2, "name": "Python for Beginners", "author": "John Doe", "cost": 150 }
]);
// Read (Find all books)
db.library.find().pretty();

// Update (Change the name of a book)
db.library.updateOne(
  { "name": "C++" },
  { $set: { "name": "Advanced C++" } }
);

// Verify the update
db.library.find({ "name": "Advanced C++" }).pretty();

// Delete (Remove a book)
db.library.deleteOne({ "bid": 2 });

// Verify the deletion
db.library.find().pretty();

// Logical Operators
// Find books with a specific author
db.library.find({ "author": "Bjarne Stroustrup" }).pretty();

// Cost-based query - Find books with cost greater than 200
db.library.find({ "cost": { $gt: 200 } }).pretty();

// Save method (to update or insert a document)
db.library.save({
  "_id": ObjectId("63721a98c2a359df01688ae0"), // Change this ID as needed
  "bid": 3,
  "name": "Data Structures",
  "author": "Alice Green",
  "cost": 250
});

// Final read to see all documents
db.library.find().pretty()
```

Output:

```
mongosh mongodb://127.0.0.1
LibraryDB> use LibraryDB;
already on db LibraryDB
LibraryDB> db.library.insertMany([
...   { "bid": 1, "name": "C++", "author": "Bjarne Stroustrup", "cost": 300 },
...   { "bid": 2, "name": "Python for Beginners", "author": "John Doe", "cost": 150 }
... ]);db.library.find().pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
    bid: 1,
    name: 'C++',
    author: 'Bjarne Stroustrup',
    cost: 300
  },
  {
    _id: ObjectId('6716032435a2e645f9c73c0d'),
    bid: 2,
    name: 'Python for Beginners',
    author: 'John Doe',
    cost: 150
  }
]
LibraryDB> db.library.updateOne(
...   { "name": "C++" },
...   { $set: { "name": "Advanced C++" } }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
LibraryDB> db.library.find({ "name": "Advanced C++" }).pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
    bid: 1,
    name: 'Advanced C++',
    author: 'Bjarne Stroustrup',
    cost: 300
  }
]
LibraryDB> db.library.deleteOne({ "bid": 2 });
{ acknowledged: true, deletedCount: 1 }
LibraryDB> db.library.find().pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
```

Activate Windows
Go to Settings to activate Windows.

```
mongosh mongodb://127.0.0.1
LibraryDB> db.library.find().pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
    bid: 1,
    name: 'Advanced C++',
    author: 'Bjarne Stroustrup',
    cost: 300
  }
]
LibraryDB> db.library.find({ "author": "Bjarne Stroustrup" }).pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
    bid: 1,
    name: 'Advanced C++',
    author: 'Bjarne Stroustrup',
    cost: 300
  }
]
LibraryDB> db.library.find({ "cost": { $gt: 200 } }).pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
    bid: 1,
    name: 'Advanced C++',
    author: 'Bjarne Stroustrup',
    cost: 300
  }
]
LibraryDB> db.library.save({
...   "_id": ObjectId('63721a98c2a359df01688ae0'), // Change this ID as needed
...   "bid": 3,
...   "name": "Data Structures",
...   "author": "Alice Green",
...   "cost": 250
... });
TypeError: db.library.save is not a function
LibraryDB> db.library.find().pretty();
[
  {
    _id: ObjectId('6716032435a2e645f9c73c0c'),
    bid: 1,
    name: 'Advanced C++',
    author: 'Bjarne Stroustrup',
    cost: 300
  }
]
LibraryDB>
```

Activate Windows
Go to Settings to activate Windows.

Assignment-10

Code:

```
// Switch to LibraryDB
use LibraryDB;
// Clear existing data (if necessary)
db.books.deleteMany({});
// Sample data insertion (only two books)
db.books.insertMany([
  { "bid": 1, "name": "JavaScript Basics", "author": "Jane Doe", "cost": 200, "category":
"Programming", "publishedYear": 2020 },
  { "bid": 2, "name": "MongoDB for Beginners", "author": "John Smith", "cost": 150, "category":
"Database", "publishedYear": 2021 }
]);
// Create an index on the author field
db.books.createIndex({ author: 1 });
// Aggregation: Group by category and calculate average cost
db.books.aggregate([
  {
    $group: {
      _id: "$category", // Group by category
      averageCost: { $avg: "$cost" }, // Calculate average cost
      totalBooks: { $sum: 1 } // Count total books in each category
    }
  },
  {
    $sort: { averageCost: 1 } // Sort by average cost in ascending order
  }
]);

// Aggregation: Count books by author
db.books.aggregate([
  {
    $group: {
      _id: "$author", // Group by author
      totalBooks: { $sum: 1 } // Count total books for each author
    }
  },
  {
    $sort: { totalBooks: -1 } // Sort by total books in descending order
  }
]);

// Example of querying using the index
db.books.find({ author: "Jane Doe" }).pretty();
```

Output:

```
mongosh mongodb://127.0.0.1:27020
LibraryDB> db.library.find().pretty();
LibraryDB> db.books.deleteMany({});
{ acknowledged: true, deletedCount: 0 }
LibraryDB> db.books.insertMany([
...   { "bid": 1, "name": "JavaScript Basics", "author": "Jane Doe", "cost": 200, "category": "Programming", "publishedYear": 2020 },
...   { "bid": 2, "name": "MongoDB for Beginners", "author": "John Smith", "cost": 150, "category": "Database", "publishedYear": 2021 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6716049335a2e645f9c73c0e'),
    '1': ObjectId('6716049335a2e645f9c73c0f')
  }
}
LibraryDB> db.books.createIndex({ author: 1 });
author_1
LibraryDB> db.books.aggregate([
...   {
...     $group: {
...       _id: "$category", // Group by category
...       averageCost: { $avg: "$cost" }, // Calculate average cost
...       totalBooks: { $sum: 1 } // Count total books in each category
...     },
...   },
...   {
...     $sort: { averageCost: 1 } // Sort by average cost in ascending order
...   }
... ]);
[
  { _id: 'Database', averageCost: 150, totalBooks: 1 },
  { _id: 'Programming', averageCost: 200, totalBooks: 1 }
]
LibraryDB> db.books.aggregate([
...   {
...     $group: {
...       _id: "$author", // Group by author
...       totalBooks: { $sum: 1 } // Count total books for each author
...     },
...   },
...   {
...     $sort: { totalBooks: -1 } // Sort by total books in descending order
...   }
... ]);
[
  { _id: 'Jane Doe', totalBooks: 1 },
  { _id: 'John Smith', totalBooks: 1 }
]
LibraryDB> db.books.find({ author: "Jane Doe" }).pretty();
```

Activate Windows
Go to Settings to activate Windows.

```
mongosh mongodb://127.0.0.1:27020
...   averageCost: { $avg: "$cost" }, // Calculate average cost
...   totalBooks: { $sum: 1 } // Count total books in each category
... },
... {
...   $sort: { averageCost: 1 } // Sort by average cost in ascending order
... }
... ]);
[
  { _id: 'Database', averageCost: 150, totalBooks: 1 },
  { _id: 'Programming', averageCost: 200, totalBooks: 1 }
]
LibraryDB> db.books.aggregate([
...   {
...     $group: {
...       _id: "$author", // Group by author
...       totalBooks: { $sum: 1 } // Count total books for each author
...     },
...   },
...   {
...     $sort: { totalBooks: -1 } // Sort by total books in descending order
...   }
... ]);
[
  { _id: 'Jane Doe', totalBooks: 1 },
  { _id: 'John Smith', totalBooks: 1 }
]
LibraryDB> db.books.find({ author: "Jane Doe" }).pretty();
[
  {
    _id: ObjectId('6716049335a2e645f9c73c0e'),
    bid: 1,
    name: 'JavaScript Basics',
    author: 'Jane Doe',
    cost: 200,
    category: 'Programming',
    publishedYear: 2020
  }
]
LibraryDB>
```

Activate Windows
Go to Settings to activate Windows.

Assignment-11

Code:

```
// Switch to LibraryDB
use LibraryDB;

// Clear existing data (if necessary)
db.books.deleteMany({});

// Sample data insertion (only two example books)
db.books.insertMany([
  { "bid": 1, "name": "JavaScript Basics", "author": "Jane Doe", "cost": 200, "category":
"Programming", "publishedYear": 2020 },
  { "bid": 2, "name": "MongoDB for Beginners", "author": "John Smith", "cost": 150, "category":
"Database", "publishedYear": 2021 }
]);

// Map function to emit the category and cost
var mapFunction = function() {
  emit(this.category, this.cost);
};

// Reduce function to sum costs for each category
var reduceFunction = function(key, values) {
  return Array.sum(values);
};

// Execute MapReduce operation
db.books.mapReduce(
  mapFunction,      // Map function
  reduceFunction,    // Reduce function
  {
    out: "category_cost", // Output collection name
    finalize: function(key, reducedValue) {
      return { totalCost: reducedValue }; // Finalize function to format output
    }
  }
);

// Display the results
db.category_cost.find().pretty();
```

Output:

```
mongosh mongodb://127.0.0.1:27017/ LibraryDB> db.books.deleteMany({});
{ acknowledged: true, deletedCount: 2 }
LibraryDB> db.books.insertMany([
...   { "bid": 1, "name": "JavaScript Basics", "author": "Jane Doe", "cost": 200, "category": "Programming", "publishedYear": 2020 },
...   { "bid": 2, "name": "MongoDB for Beginners", "author": "John Smith", "cost": 150, "category": "Database", "publishedYear": 2021 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('671605de35a2e645f9c73c10'),
    '1': ObjectId('671605de35a2e645f9c73c11')
  }
}
LibraryDB> var mapFunction = function() {
...   emit(this.category, this.cost);
... };

LibraryDB> var reduceFunction = function(key, values) {
...   return Array.sum(values);
... };

LibraryDB> db.books.mapReduce(
...   mapFunction,           // Map function
...   reduceFunction,        // Reduce function
...   {
...     out: "category_cost", // Output collection name
...     finalize: function(key, reducedValue) {
...       return { totalCost: reducedValue }; // Finalize function to format output
...     }
...   }
... );
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'category_cost', ok: 1 }
LibraryDB> db.category_cost.find().pretty();
[
  { _id: 'Programming', value: { totalCost: 200 } },
  { _id: 'Database', value: { totalCost: 150 } }
]
LibraryDB>
```