BHARATI VIDYAPEETH'S

INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT

(Affiliated to Guru Gobind Singh IndraprasthaUniversity, Approved by AICTE, New Delhi)

DATABASE MANAGEMENT SYSTEMS

(MCA-165)

PRACTICAL FILE

Submitted To:

Dr. Rakhee

(Associate Professor)

Submitted by:

Sidak Ahuja

Enrollment No.-02611604423

MCA SEM-1, SECTION-1

INDEX

		(Cid#, CName, Semester, Credits, Fee) Enrollment (Roll, Cid, DateOfReg) Now write SQL queries for the following:			
6	CP1	Consider any table of your choice and execute following functions on numeric, character and date type fields :	31-35		
7	DP1	Write a PL/SQL code that will reverse a given number like 12345 to 54321	36-38		
8	DP2	Write a PL/SQL code to accept emp_id from user and display his / her record. Table: Employee (EmpId, EName, DOJ, DOB, Salary, Address)	38-40		
9	DP3	Write a PL/SQL code to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named Areas, consisting of two columns Radius and Area. Table: Circle (Radius, Area, Circumference)	41-43		
10	DP4	Write a PL/SQL code that first withdraws an amount of Rs. 1500 from an account. Then deposits an amount of Rs. 50,000 in another account. Update the current balance. Then check to see that the sum of the current	44-46		

		balance of all the accounts in the bank does not exceed Rs. 60,00,000. If the balance exceeds Rs 60,00,000 then undo the deposit just made. Table : Accounts(AcCNo, Name, Balance, DateofOpening, DateofLastTransaction			
11	DP5	Write a PL/SQL code that accepts the account number from the user, check the users balance and updates date of last transaction. If it is greater than 1000, only then deduct Rs. 200/- from the balance, otherwise rollback the date of transaction to set it to the previous date. The process is fired on the Acct_Mstr table. Table : Acct_Mstr(AcCNo, Name, Balance, DateofOpening, DateofLastTransaction)	47-50		
12	EP1	Write a PL/SQL code that updates the balance of the account holder of 'Delhi' branch and displays the number of records updated. (Use implicit cursor) Table : Accounts(AcCNo, Name, Balance, DateofOpening, Branch)	50-51		
13	EP2	Write a Cursor(PL/SQL code) to display the Employee_name, DateofBirth,	52-55		

		Designation whose basic salary is greater than 15000, if not found then show the proper error message. (Use Exception handing) Table : Employee(EmpId, EName, DOJ, DOB, Salary, Address)			
14	FP1	Create a function that accepts emp id from the calling procedure and returns his salary. Table : Employee(EmpId, EName, DOJ, DOB, Salary, Address)	56-58		
15	FP2	Write a PL/SQL function ODDEVEN to return value TRUE if the number passed to it is EVEN else will return FALSE.	59-61		

AP-1 Consider the following table:

Table : Book

Field Name	Data type	Size	Constraint
Book_Code	Varchar2	5	Primary key
ISBN No.	Varchar2	8	Not Null/
Book_Name	Varchar2	8	Unique
Publisher	Varchar2	6	
Price	Number	5,2	>=100
Author_Name	Varchar2	8	
Date_of_Launch	Date		

Write queries to perform the following:

1. Insert few records in it.

2. Increase the size of the Book_Name field to 10 characters.

3. Display the total price of all the books of "TMH' publishing house.

4. Display the details of all the books of author "Yashwant Kanetkar" and "Robert

Lafore".

5. Delete all the books belonging to “PHI” publisher.

6. Update the price of books by 5% which belong to ‘BPB’ publishers.

7. Create a new table “Author” with author details such as author id (primary key),

name of the author, subject, contact details, research area etc.

8. Insert records in it and display the records of those authors who have ‘Web mining’ as their research area.

9. Add a new column “Author_Id” in “Book” table and make it a foreign key on “author id” of “Author” table.

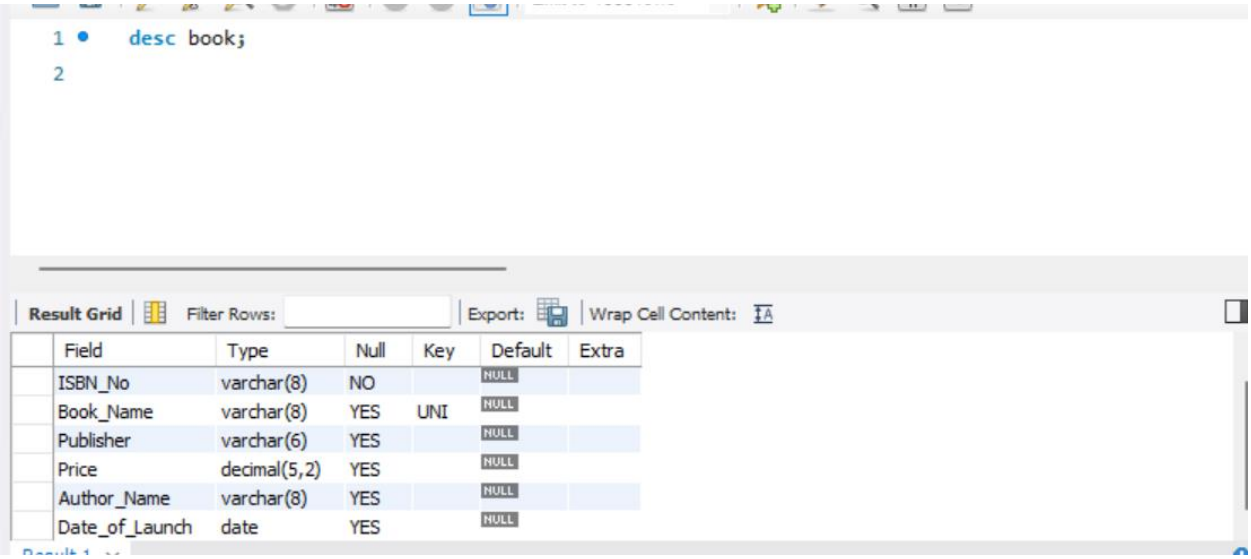
10. Update all the previous records of Book table to add information of author id (newly added column)

11. Display the records of the books where book name starts with ‘C’ or ‘G’ or ‘K’.

12. Create a view for the user to access Book Code, Name and Price of books which

are published by ‘Pearson Education’.

SOLUTIONS:



1.

```
('B001', 'ISBN001', 'Book1', 'TMH', 120.50, 'Yashwant Kanetkar', '2023-01-01'),
('B002', 'ISBN002', 'Book2', 'BPB', 190.00, 'Robert Lafore','2023-02-01'),
('B003', 'ISBN003', 'Book3', 'PHI', 150.75, 'Yashwant Kanetkar', '2023-03-01') ;

select * from book
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Book_Code	ISBN_No	Book_Name	Publisher	Price	Author_Name	Date_of_Launch
B001	ISBN001	Book1	TMH	120.50	Yashwant Kanetkar	2023-01-01
B002	ISBN002	Book2	BPB	190.00	Robert Lafore	2023-02-01
B003	ISBN003	Book3	PHI	150.75	Yashwant Kanetkar	2023-03-01
NULL	NULL	NULL	NULL	NULL	NULL	NULL

book 4 | Apply

2.

Limit to 1000 rows

```
ALTER TABLE Book MODIFY Book_Name VARCHAR(10);
```

3.

```
SELECT SUM(Price) AS Total_Price
FROM Book
WHERE Publisher = 'TMH';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_Price
120.50

4.

```
SELECT *
FROM Book
WHERE Author_Name IN ('Yashwant Kanetkar', 'Robert Lafore');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Book_Code	ISBN_No	Book_Name	Publisher	Price	Author_Name	Date_of_Launch
B001	ISBN001	Book1	TMH	120.50	Yashwant Kanetkar	2023-01-01
B002	ISBN002	Book2	BPB	190.00	Robert Lafore	2023-02-01
B003	ISBN003	Book3	PHI	150.75	Yashwant Kanetkar	2023-03-01
NULL	NULL	NULL	NULL	NULL	NULL	NULL

5.

```
1 •  DELETE FROM Book
2    WHERE Publisher = 'PHI';
3
4 •  select * from book
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conte

Book_Code	ISBN_No	Book_Name	Publisher	Price	Author_Name	Date_of_Launch
B001	ISBN001	Book1	TMH	120.50	Yashwant Kanetkar	2023-01-01
B002	ISBN002	Book2	BPB	190.00	Robert Lafore	2023-02-01
NULL	NULL	NULL	NULL	NULL	NULL	NULL

6.

```
1 •  UPDATE Book
2    SET Price = Price * 1.05
3    WHERE Publisher = 'BPB';
4 •  select* from book
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Book_Code	ISBN_No	Book_Name	Publisher	Price	Author_Name	Date_of_Launch
B001	ISBN001	Book1	TMH	120.50	Yashwant Kanetkar	2023-01-01
B002	ISBN002	Book2	BPB	199.50	Robert Lafore	2023-02-01
NULL	NULL	NULL	NULL	NULL	NULL	NULL

7.

```
CREATE TABLE Author (
    Author_Id VARCHAR(5) PRIMARY KEY,
    Author_Name VARCHAR(50),
    Subject VARCHAR(50),
    Contact_Details VARCHAR(20),
    Research_Area VARCHAR(50)
);
desc author;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Author_Id	varchar(5)	NO	PRI	NULL	
Author_Name	varchar(50)	YES		NULL	
Subject	varchar(50)	YES		NULL	
Contact_Details	varchar(20)	YES		NULL	
Research_Area	varchar(50)	YES		NULL	

8.

Limit to 1000 rows

```
INSERT INTO Author (Author_Id, Author_Name, Subject, Contact_Details, Research_Area)
VALUES
('A001', 'Author1', 'Subject1', 'Contact1', 'Web mining'),
('A002', 'Author2', 'Subject2', 'Contact2', 'Database'),
('A003', 'Author3', 'Subject3', 'Contact3', 'Web mining');

SELECT *
FROM Author
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Author_Id	Author_Name	Subject	Contact_Details	Research_Area
A001	Author1	Subject1	Contact1	Web mining
A003	Author3	Subject3	Contact3	Web mining
NULL	NULL	NULL	NULL	NULL

Result Grid
Form Editor
Field Types

9.

```
2
3 •  ALTER TABLE Book
4    ADD CONSTRAINT fk_author
5    FOREIGN KEY (Author_Id)
6    REFERENCES Author (Author_Id);
7
8 •   desc book;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Book_Code	varchar(5)	NO	PRI	NULL	
ISBN_No	varchar(8)	NO		NULL	
Book_Name	varchar(10)	YES	UNI	NULL	
Publisher	varchar(6)	YES		NULL	
Price	decimal(5,2)	YES		NULL	
Author_Name	varchar(25)	YES		NULL	
Date_of_Launch	date	YES		NULL	
Author_Id	varchar(5)	YES		NULL	

10.

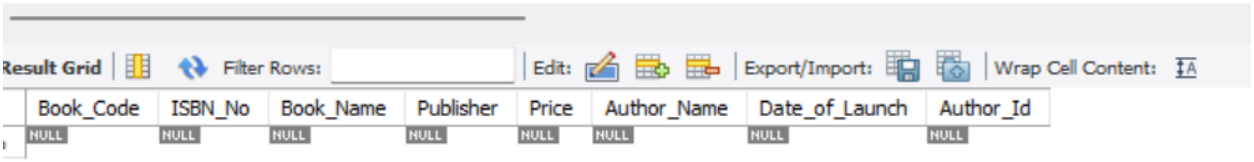
```
1 •  UPDATE Book
2    SET Author_Id = 'A001'
3    WHERE Author_Name = 'Yashwant Kanetkar';
4
5 •  Select * from book;
6
7
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

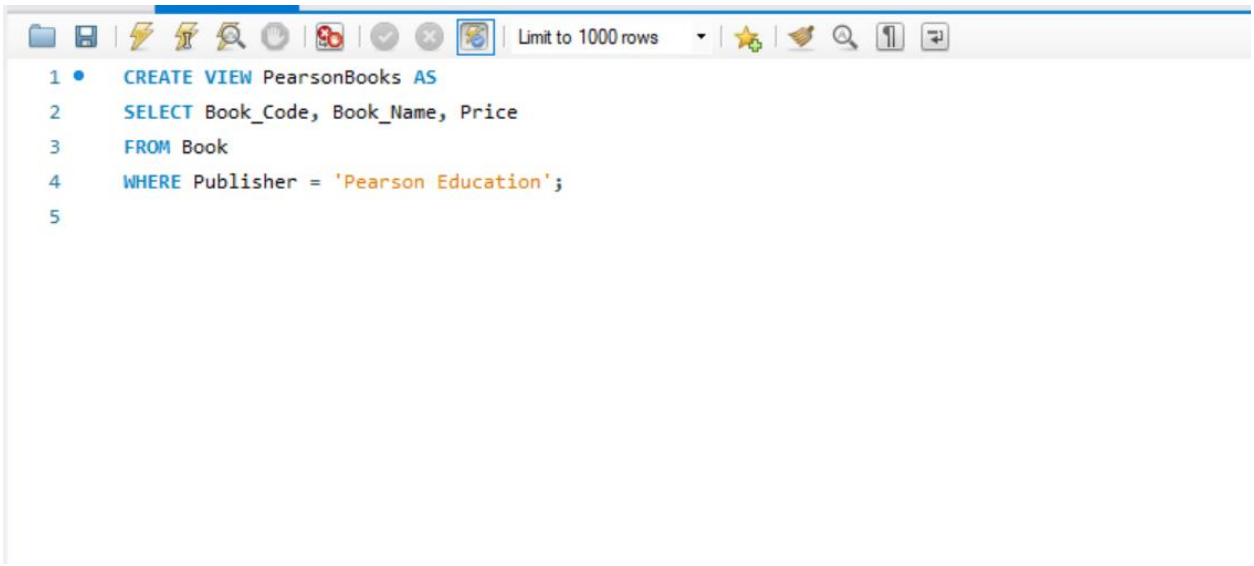
Book_Code	ISBN_No	Book_Name	Publisher	Price	Author_Name	Date_of_Launch	Author_Id
B001	ISBN001	Book1	TMH	120.50	Yashwant Kanetkar	2023-01-01	A001
B002	ISBN002	Book2	BPB	199.50	Robert Lafore	2023-02-01	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

11.

```
1 •  SELECT *
2    FROM Book
3    WHERE SUBSTR(Book_Name, 1, 1) IN ('C', 'G', 'K');
4
```



12.



AP2: Consider the following table:

Table : Student

Field Name	Data type	Size	Constraint
Roll_no	Number	2	Primary Key
Name	Varchar2	20	Not Null
Address	Varchar2	50	
City	Varchar2	30	
Marks_Sub1	Number	5,2	>=0 and <=100
Marks_sub2	Number	5,2	>=0 and <=100
Marks_sub3	Number	5,2	>=0 and <=100
Total	Number	5,2	>=0 and <=300

Write SQL queries for :

1. Insert few records except total marks.

2. Use Sequences for roll_no field.

3. Create a view on the table to access roll no, name (in capitals), Marks of all 3 subjects and Percentage of all the students.

4. Create a view on the table to access roll no, name and total marks of all the students.

5. Create a view on the table to access roll no, name and total marks of those students who belong to 'Delhi'.

6. Create a similar table 'Scholar' with same set of attributes.

7. Create a composite index on Student table.

8. Insert few records in 'Scholar' table.

9. Use 'union' to retrieve all records from both the tables.

10. Use 'intersect' to retrieve all uncommon records from both the tables.

11. Use 'minus' to retrieve records of one table which do not belong to other table.

SOLUTIONS:

1.

```
INSERT INTO Student (Roll_no, Name, Address, City, Marks_Sub1, Marks_Sub2, Marks_Sub3)
VALUES
(11, 'Sidak', 'Paschim vihar', 'delhi', 85.5, 90.0, 78.5),
(21, 'Lakshya', '456 Oak St', 'delhi', 75.0, 88.5, 92.0),
(31, 'Ishaan', '789 Pine St', 'himachal pradesh', 92.5, 85.0, 89.5);

select * from student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Roll_no	Name	Address	City	Marks_Sub1	Marks_Sub2	Marks_Sub3	Total
1	Sidak	123 Main St	New York	85.50	90.00	78.50	NULL
2	Lakshya	456 Oak St	Los Angeles	75.00	88.50	92.00	NULL
3	Ishaan	789 Pine St	Chicago	92.50	85.00	89.50	NULL
11	Sidak	Paschim vihar	delhi	85.50	90.00	78.50	NULL
21	Lakshya	456 Oak St	delhi	75.00	88.50	92.00	NULL
31	Ishaan	789 Pine St	himachal pradesh	92.50	85.00	89.50	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2.

```
ALTER TABLE `practical`.`student`
CHANGE COLUMN `Roll_no` `Roll_no` INT NOT NULL AUTO_INCREMENT ;

```

```sql
INSERT INTO Student ( Name, Address, City, Marks_Sub1, Marks_Sub2, Marks_Sub3)
VALUES
( 'Tushar', 'Sulatanpuri', 'Delhi', 85.5, 90.0, 78.5),
( 'Pratishtha', 'Indirapuram', 'Uttar Pradesh', 75.0, 88.5, 92.0),
( 'Srishti', 'Delhi', 'Delhi', 92.5, 85.0, 89.5);

select * from student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Roll_no	Name	Address	City	Marks_Sub1	Marks_Sub2	Marks_Sub3	Total
1	Sidak	123 Main St	New York	85.50	90.00	78.50	NULL
2	Lakshya	456 Oak St	Los Angeles	75.00	88.50	92.00	NULL
3	Ishaan	789 Pine St	Chicago	92.50	85.00	89.50	NULL
11	Sidak	Paschim vihar	delhi	85.50	90.00	78.50	NULL
21	Lakshya	456 Oak St	delhi	75.00	88.50	92.00	NULL
31	Ishaan	789 Pine St	himachal pradesh	92.50	85.00	89.50	NULL
32	Tushar	Sulatanpuri	Delhi	85.50	90.00	78.50	NULL
33	Pratishtha	Indirapuram	Uttar Pradesh	75.00	88.50	92.00	NULL
34	Srishti	Delhi	Delhi	92.50	85.00	89.50	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3.

```sql
CREATE VIEW StudentView AS
SELECT
   Roll_no,
   UPPER(Name) AS Name,
   Marks_Sub1, Marks_Sub2, Marks_Sub3,
   ((Marks_Sub1 + Marks_Sub2 + Marks_Sub3) / 300) * 100 AS Percentage
FROM Student;
```

```
8 •    select * from studentview;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Roll_no	Name	Marks_Sub1	Marks_Sub2	Marks_Sub3	Percentage
1	SIDAK	85.50	90.00	78.50	84.666667
2	LAKSHYA	75.00	88.50	92.00	85.166667
3	ISHAAN	92.50	85.00	89.50	89.000000
11	SIDAK	85.50	90.00	78.50	84.666667
21	LAKSHYA	75.00	88.50	92.00	85.166667
31	ISHAAN	92.50	85.00	89.50	89.000000
32	TUSHAR	85.50	90.00	78.50	84.666667
33	PRATISHTHA	75.00	88.50	92.00	85.166667
34	SRISHTI	92.50	85.00	89.50	89.000000

4.

```
1 •    CREATE VIEW StudentTotalView AS
2      SELECT Roll_no, Name, Total
3      FROM Student;
4
5 •    select * from studenttotalview;
6
```

esult Grid | Filter Rows: | Export: | Wrap Ce

Roll_no	Name	Total
1	Sidak	NULL
2	Lakshya	NULL
3	Ishaan	NULL
11	Sidak	NULL
21	Lakshya	NULL
31	Ishaan	NULL
32	Tushar	NULL
33	Pratishtha	NULL
34	Srishti	NULL

5.

```sql
CREATE VIEW StudentDelhiView AS
SELECT Roll_no, Name, Total
FROM Student
WHERE City = 'Delhi';

select * from studentdelhiview;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Roll_no	Name	Total
11	Sidak	NULL
21	Lakshya	NULL
32	Tushar	NULL
34	Srishti	NULL

6.

```sql
CREATE TABLE Scholar (
    Roll_no int(2) PRIMARY KEY,
    Name Varchar(20) NOT NULL,
    Address Varchar(50),
    City Varchar(30),
    Marks_Sub1 decimal(5,2) CHECK (Marks_Sub1 >= 0 AND Marks_Sub1 <= 100),
    Marks_sub2 decimal(5,2) CHECK (Marks_Sub2 >= 0 AND Marks_Sub2 <= 100),
    Marks_sub3 decimal(5,2) CHECK (Marks_Sub3 >= 0 AND Marks_Sub3 <= 100),
    Total decimal(5,2) CHECK (Total >= 0 AND Total <= 300)
);

```

7.

```sql
CREATE INDEX idx_student_composite ON Student (City, Total);

 show index from student;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Ir
student	0	PRIMARY	1	Roll_no	A	3	NULL	NULL		BT
student	1	idx_student_composite	1	City	A	5	NULL	NULL	YES	BT
student	1	idx_student_composite	2	Total	A	5	NULL	NULL	YES	BT

8.

Limit to 1000 rows

```sql
INSERT INTO Scholar (Roll_no, Name, Address, City, Marks_Sub1, Marks_Sub2, Marks_Sub3)
VALUES
(101, 'Arjun', '321 Pine St', 'Delhi', 95.0, 88.5, 92.5),
(102, 'Garima', '654 Maple St', 'Mumbai', 80.0, 92.5, 87.0),
(103, 'Lakshya', '987 Cedar St', 'Kolkata', 78.5, 85.0, 90.5);

```

9.

```
1 •  SELECT * FROM Student
2    UNION
3    SELECT * FROM Scholar;
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Roll_no	Name	Address	City	Marks_Sub1	Marks_Sub2	Marks_Sub3	Total
1	Sidak	123 Main St	New York	85.50	90.00	78.50	NULL
2	Lakshya	456 Oak St	Los Angeles	75.00	88.50	92.00	NULL
3	Ishaan	789 Pine St	Chicago	92.50	85.00	89.50	NULL
11	Sidak	Paschim vihar	delhi	85.50	90.00	78.50	NULL
21	Lakshya	456 Oak St	delhi	75.00	88.50	92.00	NULL
31	Ishaan	789 Pine St	himachal pradesh	92.50	85.00	89.50	NULL
32	Tushar	Sulatanpuri	Delhi	85.50	90.00	78.50	NULL
33	Pratishtha	Indirapuram	Uttar Pradesh	75.00	88.50	92.00	NULL
34	Srishti	Delhi	Delhi	92.50	85.00	89.50	NULL
101	Arjun	321 Pine St	Delhi	95.00	88.50	92.50	NULL
102	Garima	654 Maple St	Mumbai	80.00	92.50	87.00	NULL
103	Lakshya	987 Cedar St	Kolkata	78.50	85.00	90.50	NULL

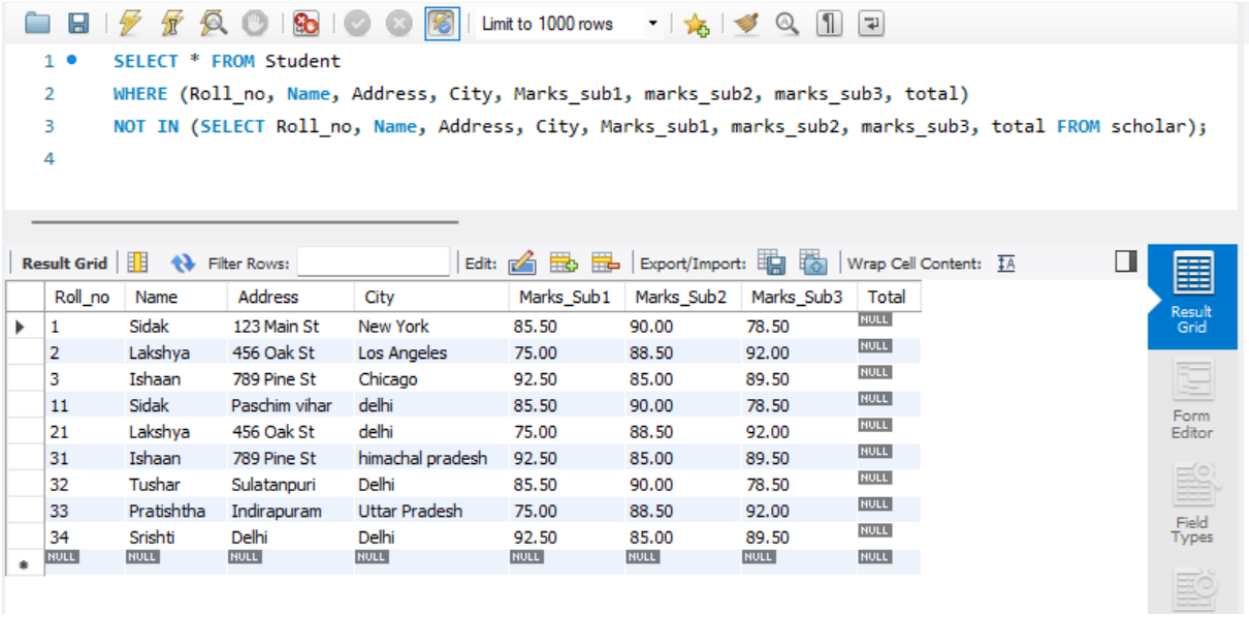
10.

```
SELECT * FROM Student
INTERSECT
SELECT * FROM Scholar;
```

Filter Rows: | Export: | Wrap Cell Content:

no	Name	Address	City	Marks_Sub1	Marks_Sub2	Marks_Sub3	Total

11.



Roll_no	Name	Address	City	Marks_Sub1	Marks_Sub2	Marks_Sub3	Total
1	Sidak	123 Main St	New York	85.50	90.00	78.50	NULL
2	Lakshya	456 Oak St	Los Angeles	75.00	88.50	92.00	NULL
3	Ishaan	789 Pine St	Chicago	92.50	85.00	89.50	NULL
11	Sidak	Paschim vihar	delhi	85.50	90.00	78.50	NULL
21	Lakshya	456 Oak St	delhi	75.00	88.50	92.00	NULL
31	Ishaan	789 Pine St	himachal pradesh	92.50	85.00	89.50	NULL
32	Tushar	Sulatanpuri	Delhi	85.50	90.00	78.50	NULL
33	Pratishtha	Indirapuram	Uttar Pradesh	75.00	88.50	92.00	NULL
34	Srishti	Delhi	Delhi	92.50	85.00	89.50	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

AA1 Consider the following table:

Table : Student

Field Name	Data type	Size	Constraint
Roll_no	Number	2	Primary Key
Name	Varchar2	20	Not Null
Address	Varchar2	50	
City	Varchar2	30	
Marks_Sub1	Number	5,2	>=0 and <=100
Marks_sub2	Number	5,2	>=0 and <=100
Marks_sub3	Number	5,2	>=0 and <=100
Total	Number	5,2	>=0 and <=300

Write sql queries for:

1. Insert few records except total marks.

2. Create a view on the table to access roll no, name and total marks of those students who belong to 'Delhi'.

3. Calculate total marks of each student and save it.

4. Display the name of the student who has got the highest marks.

5. Display the records of those students who have got marks more than the marks of

‘Kiran’.

6. Display the percentage of all the students with their names.

7. Display the names of those students who have got maximum marks in sub1 and

more than 50 in sub2 and sub3.

8. Update the marks of all those students who are fail, with grace marks 5. (fail means less than 40)

9. Delete the record of student who got minimum marks.

10. Display record of student who got second highest marks.

11. Display the average total marks of all the students city wise excluding the students of ‘Gurgaon’.

12. Get the names of students with Roll_no less than 30 and whose total marks is

more than the total marks of at least one student with Roll_no greater than or equal to 30.

SOLUTIONS:

1.

```
mysql> select * from students;
+---------+--------+---------+---------+------------+------------+------------+-------+
| roll_no | name   | address | city    | marks_sub1 | marks_sub2 | marks_sub3 | total |
+---------+--------+---------+---------+------------+------------+------------+-------+
|       1 | ayushi | UP      | meerut  |         98 |         99 |         97 |  NULL |
|       2 | shruti | UP      | mzn     |         94 |         96 |         97 |  NULL |
|       3 | vishal | UP      | noida   |         94 |         96 |         95 |  NULL |
|       4 | saumya | UP      | gzb     |         94 |         94 |         93 |  NULL |
|       5 | vansh  | UP      | baghpat |         94 |         94 |         91 |  NULL |
+---------+--------+---------+---------+------------+------------+------------+-------+
```

2.

```
mysql> select roll_no,name,total from students where city='delhi';
+---------+-------+-------+
| roll_no | name  | total |
+---------+-------+-------+
|       5 | rohan |   278 |
+---------+-------+-------+
```

3.

```
mysql> select roll_no,name,address,city,marks_sub1,marks_sub2,marks_sub3,(marks_sub1+marks_sub2+marks_sub3)as total from students;
+---------+--------+---------+---------+-----------+-----------+-----------+-------+
| roll_no | name   | address | city    | marks_sub1 | marks_sub2 | marks_sub3 | total |
+---------+--------+---------+---------+-----------+-----------+-----------+-------+
|       1 | ayushi | UP      | meerut  |         98 |         99 |         97 |   294 |
|       2 | shruti | UP      | mzn     |         94 |         96 |         97 |   287 |
|       3 | vishal | UP      | noida   |         94 |         96 |         95 |   285 |
|       4 | saumya | UP      | gzb     |         94 |         94 |         93 |   281 |
|       5 | vansh  | UP      | baghpat |         94 |         94 |         91 |   279 |
|       6 | rohan  | DELHI   | delhi   |         94 |         92 |         92 |   278 |
+---------+--------+---------+---------+-----------+-----------+-----------+-------+
```

4.

```
mysql> select * from students where total= (select max(total) from students);
+---------+--------+---------+--------+-----------+-----------+-----------+-------+
| roll_no | name   | address | city   | marks_sub1 | marks_sub2 | marks_sub3 | total |
+---------+--------+---------+--------+-----------+-----------+-----------+-------+
|       1 | ayushi | UP      | meerut |         98 |         99 |         97 |   294 |
+---------+--------+---------+--------+-----------+-----------+-----------+-------+
```

5.

```
mysql> select * from students where total>(select total from students where name='kiran');
+---------+--------+---------+--------+-----------+-----------+-----------+-------+
| roll_no | name   | address | city   | marks_sub1 | marks_sub2 | marks_sub3 | total |
+---------+--------+---------+--------+-----------+-----------+-----------+-------+
|       1 | ayushi | UP      | meerut |         98 |         99 |         97 |   294 |
|       2 | shruti | UP      | mzn    |         94 |         96 |         97 |   287 |
|       3 | vishal | UP      | noida  |         94 |         96 |         95 |   285 |
+---------+--------+---------+--------+-----------+-----------+-----------+-------+
```

6.

```
mysql> select name,total/3 from students;
+--------+----------+
| name   | total/3  |
+--------+----------+
| ayushi | 98.0000  |
| shruti | 95.6667  |
| vishal | 95.0000  |
| vansh  | 93.0000  |
| rohan  | 92.6667  |
| kiran  | 94.6667  |
+--------+----------+
6 rows in set (0.00 sec)
```

7.

```
mysql> select name from students where marks_sub1 =(select max(marks_sub1)from students) and marks_sub2 >50 and marks_sub3>50;
+--------+
| name   |
+--------+
| ayushi |
| kiran  |
+--------+
```

8.

```
mysql> update students set marks_sub1= marks_sub1+5, marks_sub2 = marks_sub2+5, marks_sub3 = marks_sub3+5 where marks_sub1<40 or marks_sub2<40 or marks_sub3
<40;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from students;
+---------+--------+-------------+---------+------------+------------+------------+-------+
| roll_no | name   | address     | city    | marks_sub1 | marks_sub2 | marks_sub3 | total |
+---------+--------+-------------+---------+------------+------------+------------+-------+
|       1 | ayushi | UP          | meerut  |         98 |         99 |         97 |   294 |
|       2 | shruti | UP          | mzn     |         94 |         96 |         97 |   287 |
|       3 | vishal | UP          | noida   |         94 |         96 |         95 |   285 |
|       4 | vansh  | UP          | baghpat |         94 |         94 |         91 |   279 |
|       5 | rohan  | DELHI       | delhi   |         94 |         93 |         91 |   278 |
|       6 | kiran  | maharashtra | mumbai  |         98 |         92 |         94 |   284 |
|       7 | mike   | maharashtra | mumbai  |         45 |         43 |         28 |   101 |
+---------+--------+-------------+---------+------------+------------+------------+-------+
```

9.

```
mysql> select * from students;
+---------+--------+-------------+---------+------------+------------+------------+-------+
| roll_no | name   | address     | city    | marks_sub1 | marks_sub2 | marks_sub3 | total |
+---------+--------+-------------+---------+------------+------------+------------+-------+
|       1 | ayushi | UP          | meerut  |         98 |         99 |         97 |   294 |
|       2 | shruti | UP          | mzn     |         94 |         96 |         97 |   287 |
|       3 | vishal | UP          | noida   |         94 |         96 |         95 |   285 |
|       4 | vansh  | UP          | baghpat |         94 |         94 |         91 |   279 |
|       5 | rohan  | DELHI       | delhi   |         94 |         93 |         91 |   278 |
|       7 | mike   | maharashtra | mumbai  |         45 |         43 |         28 |   101 |
+---------+--------+-------------+---------+------------+------------+------------+-------+
```

10.

```
mysql> SELECT *
    -> FROM Student
    -> ORDER BY Total DESC
    -> LIMIT 1 OFFSET 1;
+---------+-----------+----------+-------+------------+------------+------------+--------+
| Roll_no | Name      | Address  | City  | Marks_Sub1 | Marks_Sub2 | Marks_Sub3 | Total  |
+---------+-----------+----------+-------+------------+------------+------------+--------+
|      12 | Divyanshi | C-34 NSP | Delhi |      95.00 |      87.00 |     100.00 | 282.00 |
+---------+-----------+----------+-------+------------+------------+------------+--------+
1 row in set (0.00 sec)
```

11.

students of ‘Gurgaon’.

```
here city<> gurgaon ' at line 1
mysql> select city,AVG(total) as 'average_marks' from students where city<>'gurgaon' GROUP BY roll_no ;
+---------+---------------+
| city    | average_marks |
+---------+---------------+
| meerut  |      294.0000 |
| mzn     |      287.0000 |
| noida   |      285.0000 |
| baghpat |      279.0000 |
| delhi   |      278.0000 |
| mumbai  |      101.0000 |
+---------+---------------+
```

12.

```
mysql> select name from students where roll_no<3 and total>=(select min(total)from students where roll_no>3);
+--------+
| name   |
+--------+
| ayushi |
| shruti |
+--------+
2 rows in set (0.00 sec)
```

BP1: Consider the following three relations:

Primary Keys are indicated by #

Foriegn Keys are underlined.

Employee (Eid#, EName, Address, DateOfJoining, Department)

Project (Pid#, PName, StartDate, TerminationDate)

AssignedTo (Eid, Pid)

Now write queries for the following in relational algebra as well as in SQL:

1. Find the employees working on 'Banking' project.

2. Find the projects assigned to the employees of D01 or D02.

3. Find the employees who belong to Delhi and work on either 'University' project or 'ShareMarket' project

4. Find the employees who do not work on 'University' project.

5. Find the employees who work on all projects.

6. List the employees who have not been assigned any project.

7. Find the employees who joined the department after the commencement of 'Bank' project.

8. Display the start and termination date of projects which are allotted to 'Jai Prakash'.

RELATIONAL ALGEBRA

1.

$$\pi_{Eid,EName}(Employee \bowtie_{Eid=Eid} \sigma_{PName='Banking'}(Project \bowtie_{Pid=Pid} AssignedTo))$$

2.

$$\pi_{Pid,PName}(Project \bowtie_{Pid=Pid} (\sigma_{Department='D01' \vee Department='D02'}(Employee) \bowtie_{Eid=Eid} AssignedTo))$$

3.

$$\pi_{Eid,EName}(Employee \bowtie_{Eid=Eid} (\sigma_{City='Delhi'}(Employee) \bowtie_{Eid=Eid} AssignedTo \bowtie_{Pid=Pid} \sigma_{PName='University' \vee PName='ShareMarket'}(Project)))$$

4.

$$\pi_{Eid,EName}(Employee \bowtie_{Eid=Eid} (\pi_{Eid}(AssignedTo) - \pi_{Eid}(AssignedTo \bowtie_{Pid=Pid} \sigma_{PName='University'}(Project))))$$

5.

$$\pi_{Eid,EName}(Employee \bowtie_{Eid=Eid} (\pi_{Eid,Pid}(AssignedTo) = \pi_{Eid}(AssignedTo \bowtie_{Pid=Pid} Project))$$

6.

$$\pi_{Eid,EName}(Employee - \pi_{Eid}(AssignedTo))$$

7.

$$\pi_{Eid,EName}(Employee \bowtie_{Eid=Eid} (\sigma_{DateOfJoining>StartDate}(Employee \bowtie_{Eid=Eid} AssignedTo \bowtie_{Pid=Pid} \sigma_{PName='Bank'}(Project))))$$

8.

$$\pi_{PName,StartDate,TerminationDate}(Project \bowtie_{Pid=Pid} (\sigma_{EName='JaiPrakash'}(Employee) \bowtie_{Eid=Eid} AssignedTo))$$

SQL:

```
mysql> create table employee(
    -> Eid int primary key,
    -> EName varchar(50),
    -> Address varchar(50),
    -> DateOfJoining date,
    -> Department varchar(50));
Query OK, 0 rows affected (0.03 sec)

mysql> create table project(
    -> Pid int primary key,
    -> PName varchar(50),
    -> StartDate date,
    -> TerminationDate date);
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> select * from employee;
+-----+-------+---------+---------------+------------+
| Eid | EName | Address | DateOfJoining | Department |
+-----+-------+---------+---------------+------------+
|   1 | Tom   | xyz     | 2023-01-11    | Finance    |
|   2 | Jack  | abc     | 2023-02-13    | Accounting |
|   3 | Ruby  | pqr     | 2023-03-16    | Management |
|   4 | Gwen  | def     | 2023-04-17    | Science    |
+-----+-------+---------+---------------+------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from project;
+-----+-----------------+------------+-----------------+
| Pid | PName           | StartDate  | TerminationDate |
+-----+-----------------+------------+-----------------+
|   1 | Risk Management | 2023-01-11 | 2023-02-11      |
|   2 | Stocks          | 2023-01-09 | 2023-03-16      |
|   3 | WorkForce       | 2023-03-12 | 2023-05-14      |
|   4 | AI              | 2023-03-19 | 2023-07-12      |
+-----+-----------------+------------+-----------------+
4 rows in set (0.00 sec)
```

1.

```
SELECT E.Eid, E.EName
FROM Employee E
JOIN AssignedTo A ON E.Eid = A.Eid
JOIN Project P ON A.Pid = P.Pid
WHERE P.PName = 'Banking';
```

2.

```
SELECT P.Pid, P.PName
FROM Project P
JOIN AssignedTo A ON P.Pid = A.Pid
JOIN Employee E ON A.Eid = E.Eid
WHERE E.Department IN ('D01', 'D02');
```

3.
```
SELECT E.Eid, E.EName
FROM Employee E
JOIN AssignedTo A ON E.Eid = A.Eid
JOIN Project P ON A.Pid = P.Pid
WHERE E.City = 'Delhi' AND (P.PName = 'University' OR P.PName = 'ShareMarket');
```

4.
```
SELECT E.Eid, E.EName
```

```
FROM Employee E
LEFT JOIN AssignedTo A ON E.Eid = A.Eid
LEFT JOIN Project P ON A.Pid = P.Pid
WHERE P.PName IS NULL OR P.PName <> 'University';

5.SELECT E.Eid, E.EName
FROM Employee E
WHERE NOT EXISTS (
  SELECT P.Pid
  FROM Project P
  WHERE NOT EXISTS (
    SELECT A.Pid
    FROM AssignedTo A
    WHERE A.Eid = E.Eid AND A.Pid = P.Pid
  )
);

6. SELECT E.Eid, E.EName
FROM Employee E
WHERE NOT EXISTS (
  SELECT A.Eid
  FROM AssignedTo A
  WHERE A.Eid = E.Eid
);
```

7. SELECT E.Eid, E.EName

FROM Employee E

JOIN AssignedTo A ON E.Eid = A.Eid

JOIN Project P ON A.Pid = P.Pid

WHERE E.DateOfJoining > P.StartDate AND P.PName = 'Bank';

8. SELECT P.PName, P.StartDate, P.TerminationDate

FROM Project P

JOIN AssignedTo A ON P.Pid = A.Pid

JOIN Employee E ON A.Eid = E.Eid

WHERE E.EName = 'Jai Prakash';

BA1: Consider the following relations:

Primary Keys are indicated by #

Foriegn Keys are underlined.

Student (Roll#, SName, City, Age, Gender)

Course (Cid#, CName, Semester, Credits, Fee)

Enrollment (Roll, Cid, DateOfReg)

Now write SQL queries for the following:

1. Find the details of students registered in MCA course.

2. Display the total number of students enrolled in MBA.

3. Display the names of students who are enrolled in the course in which "Anita" is registered.

4. Display the names of all the students with their date of enrollment in the

courses.

5. Display the name of the course in which “Kunal” is registered.

6. Display the names of boys-students enrolled in MCA.

7. Display the number of students registered in MCA course who live outside Delhi

SOLUTIONS:

1. SELECT *

FROM Student s

JOIN Enrollment e ON s.Roll# = e.Roll

JOIN Course c ON e.Cid = c.Cid#

WHERE c.CName = 'MCA';

2. SELECT COUNT(*) AS TotalStudentsEnrolled

FROM Student s

JOIN Enrollment e ON s.Roll# = e.Roll

JOIN Course c ON e.Cid = c.Cid#

WHERE c.CName = 'MBA';

3. SELECT DISTINCT s.SName

FROM Student s

JOIN Enrollment e ON s.Roll# = e.Roll

WHERE e.Cid IN (SELECT Cid FROM Enrollment WHERE Roll = (SELECT Roll# FROM Student WHERE SName = 'Anita'));

4. SELECT s.SName, e.DateOfReg

FROM Student s

JOIN Enrollment e ON s.Roll# = e.Roll;

5. SELECT c.CName

FROM Course c

JOIN Enrollment e ON c.Cid# = e.Cid

JOIN Student s ON e.Roll = s.Roll#

WHERE s.SName = 'Kunal';

6. SELECT s.SName

FROM Student s

JOIN Enrollment e ON s.Roll# = e.Roll

JOIN Course c ON e.Cid = c.Cid#

WHERE c.CName = 'MCA' AND s.Gender = 'Male';

7. SELECT COUNT(*) AS NumStudentsOutsideDelhi

FROM Student s

JOIN Enrollment e ON s.Roll# = e.Roll

JOIN Course c ON e.Cid = c.Cid#

WHERE c.CName = 'MCA' AND s.City <> 'Delhi';

CP1: Consider any table of your choice and execute following functions on numeric, character

and date type fields :

- **sqrt( )**

```
mysql> select sqrt(age) as age_sqrt from Class_students;
+------------------+
| age_sqrt         |
+------------------+
|  4.58257569495584 |
|  4.69041575982343 |
| 4.358898943540674 |
|  4.58257569495584 |
|  4.47213595499958 |
|  4.69041575982343 |
|  4.69041575982343 |
|  4.47213595499958 |
|  4.58257569495584 |
|  4.58257569495584 |
+------------------+
10 rows in set (0.00 sec)
```

- **power( )**

```
mysql>
mysql> select power (age, 2) as age_squared from Class_students;
+-------------+
| age_squared |
+-------------+
|         441 |
|         484 |
|         361 |
|         441 |
|         400 |
|         484 |
|         484 |
|         400 |
|         441 |
|         441 |
+-------------+
10 rows in set (0.00 sec)
```

- **ceil( )**

```
mysql> select ceil(sqrt(age)) as ceil from Class_students;
+------+
| ceil |
+------+
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
+------+
10 rows in set (0.00 sec)
```

- **max( )**

```
mysql> select max(age) as max_age from Class_students;
+---------+
| max_age |
+---------+
|      22 |
+---------+
1 row in set (0.00 sec)
```

- **min( )**

```
mysql> select min(age) as min_age from Class_students;
+---------+
| min_age |
+---------+
|      19 |
+---------+
1 row in set (0.00 sec)
```

- **avg( )**

```
mysql> select avg(age) as avg_age from Class_students;
+---------+
| avg_age |
+---------+
| 20.9000 |
+---------+
1 row in set (0.00 sec)
```

- **count( )**

```
mysql> select count(age) as item_count from Class_students;
+------------+
| item_count |
+------------+
|         10 |
+------------+
1 row in set (0.00 sec)
```

- **exp( )**

```
mysql> select exp(age) as exp_age from Class_students;
+--------------------+
| exp_age            |
+--------------------+
| 1318815734.4832146 |
|  3584912846.131592 |
| 178482300.96318725 |
| 1318815734.4832146 |
|  485165195.4097903 |
|  3584912846.131592 |
|  3584912846.131592 |
|  485165195.4097903 |
| 1318815734.4832146 |
| 1318815734.4832146 |
+--------------------+
10 rows in set (0.00 sec)
```

- **mod( )**

```
mysql> select mod(age, 12) as mod_age from Class_students;
+---------+
| mod_age |
+---------+
|       9 |
|      10 |
|       7 |
|       9 |
|       8 |
|      10 |
|      10 |
|       8 |
|       9 |
|       9 |
+---------+
10 rows in set (0.00 sec)
```

- **sum ( )**

```
mysql> select sum(age) as total_age from Class_students;
+-----------+
| total_age |
+-----------+
|       209 |
+-----------+
1 row in set (0.00 sec)
```

- **floor( )**

```
mysql> select floor(sqrt(age)) as floor from Class_students;
+-------+
| floor |
+-------+
|     4 |
|     4 |
|     4 |
|     4 |
|     4 |
|     4 |
|     4 |
|     4 |
|     4 |
|     4 |
+-------+
10 rows in set (0.00 sec)
```

- **abs( )**

```
mysql> select abs(age) as absolute_age from Class_students;
+--------------+
| absolute_age |
+--------------+
|           21 |
|           22 |
|           19 |
|           21 |
|           20 |
|           22 |
|           22 |
|           20 |
|           21 |
|           21 |
+--------------+
10 rows in set (0.00 sec)
```

- **greatest ( )**

```
mysql> select greatest(age, 20) as greatest_age from Class_students;
+--------------+
| greatest_age |
+--------------+
|           21 |
|           22 |
|           20 |
|           21 |
|           20 |
|           22 |
|           22 |
|           20 |
|           21 |
|           21 |
+--------------+
10 rows in set (0.00 sec)
```

- **round( )**

```
mysql> select round(sqrt(age), 2) as rounded_sqrt from Class_students;
+--------------+
| rounded_sqrt |
+--------------+
|         4.58 |
|         4.69 |
|         4.36 |
|         4.58 |
|         4.47 |
|         4.69 |
|         4.69 |
|         4.47 |
|         4.58 |
|         4.58 |
+--------------+
10 rows in set (0.00 sec)
```

- **to_date( )**

```
mysql> select str_to_date("August,5,2017", "%M %e %Y");
+-----------------------------------------+
| str_to_date("August,5,2017", "%M %e %Y") |
+-----------------------------------------+
| NULL                                    |
+-----------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

- **months_between( )**

```
mysql> select last_day('2023-01-01');
+------------------------+
| last_day('2023-01-01') |
+------------------------+
| 2023-01-31             |
+------------------------+
1 row in set (0.00 sec)
```

- **last_day( )**

```
mysql> select last_day('2023-01-01');
+------------------------+
| last_day('2023-01-01') |
+------------------------+
| 2023-01-31             |
+------------------------+
1 row in set (0.00 sec)
```

- **sysdate**

```
mysql> select sysdate();
+---------------------+
| sysdate()           |
+---------------------+
| 2023-12-09 19:00:49 |
+---------------------+
1 row in set (0.00 sec)
```

- **length( )**

```
mysql> select length(sname) as name_length from Class_students;
+-------------+
| name_length |
+-------------+
|           9 |
|           5 |
|           8 |
|           6 |
|           7 |
|           7 |
|           6 |
|           7 |
|           6 |
|           6 |
+-------------+
10 rows in set (0.00 sec)
```

- **substr( )**

```
mysql> select substr(sname, 1, 1) as first_letter from Class_students;
+--------------+
| first_letter |
+--------------+
| D            |
| S            |
| H            |
| I            |
| B            |
| P            |
| A            |
| S            |
| S            |
| N            |
+--------------+
10 rows in set (0.00 sec)
```

- **lower( )**

```
mysql> select lower(sname) as lowercase_name from Class_students;
+----------------+
| lowercase_name |
+----------------+
| divyanshi      |
| shrey          |
| harshita       |
| ishika         |
| bhumika        |
| prateek        |
| ayushi         |
| saksham        |
| shipra         |
| nikhil         |
+----------------+
10 rows in set (0.00 sec)
```

- **instr( )**

```
mysql> select instr(sname, 'a') as first_occurance_of_a from Class_student
s;
+----------------------+
| first_occurance_of_a |
+----------------------+
|                    5 |
|                    0 |
|                    2 |
|                    6 |
|                    7 |
|                    3 |
|                    1 |
|                    2 |
|                    6 |
|                    0 |
+----------------------+
10 rows in set (0.00 sec)
```

- **concat( )**

```
mysql> select concat(sname, " says hello!") as greeting from Class_student
s;
+----------------------+
| greeting             |
+----------------------+
| Divyanshi says hello! |
| Shrey says hello!     |
| Harshita says hello!  |
| Ishika says hello!    |
| Bhumika says hello!   |
| Prateek says hello!   |
| Ayushi says hello!    |
| Saksham says hello!   |
| Shipra says hello!    |
| Nikhil says hello!    |
+----------------------+
10 rows in set (0.00 sec)
```

- **lpad( )**

```
mysql> select lpad(sname, 20, "X") as left_padding from Class_students;
+----------------------+
| left_padding         |
+----------------------+
| XXXXXXXXXXDivyanshi |
| XXXXXXXXXXXXXXShrey |
| XXXXXXXXXXXHarshita |
| XXXXXXXXXXXXXIshika |
| XXXXXXXXXXXXBhumika |
| XXXXXXXXXXXXPrateek |
| XXXXXXXXXXXXXAyushi |
| XXXXXXXXXXXXSaksham |
| XXXXXXXXXXXXXShipra |
| XXXXXXXXXXXXXNikhil |
+----------------------+
10 rows in set (0.00 sec)
```

- **rpad( )**

```
mysql> select rpad(sname, 20, "X") as right_padding from Class_students;
+----------------------+
| right_padding        |
+----------------------+
| DivyanshiXXXXXXXXXX  |
| ShreyXXXXXXXXXXXXXX  |
| HarshitaXXXXXXXXXXX  |
| IshikaXXXXXXXXXXXXX  |
| BhumikaXXXXXXXXXXXX  |
| PrateekXXXXXXXXXXXX  |
| AyushiXXXXXXXXXXXXX  |
| SakshamXXXXXXXXXXXX  |
| ShipraXXXXXXXXXXXXX  |
| NikhilXXXXXXXXXXXXX  |
+----------------------+
10 rows in set (0.00 sec)
```

- **ltrim( )**

```
mysql> select ltrim(sname) as left_trim from Class_students;
+-----------+
| left_trim |
+-----------+
| Divyanshi |
| Shrey     |
| Harshita  |
| Ishika    |
| Bhumika   |
| Prateek   |
| Ayushi    |
| Saksham   |
| Shipra    |
| Nikhil    |
+-----------+
10 rows in set (0.00 sec)
```

- **rtrim( )**

```
mysql> select rtrim(sname) as right_trim from Class_students;
+------------+
| right_trim |
+------------+
| Divyanshi  |
| Shrey      |
| Harshita   |
| Ishika     |
| Bhumika    |
| Prateek    |
| Ayushi     |
| Saksham    |
| Shipra     |
| Nikhil     |
+------------+
10 rows in set (0.00 sec)
```

- **replace( )**

```
mysql> select replace(sname, 'a', "Alien") as Alien from Class_students;
+------------------+
| Alien            |
+------------------+
| DivyAliennshi    |
| Shrey            |
| HAlienrshitAlien |
| IshikAlien       |
| BhumikAlien      |
| PrAlienteek      |
| Ayushi           |
| SAlienkshAlienm  |
| ShiprAlien       |
| Nikhil           |
+------------------+
10 rows in set (0.00 sec)
```

DP1: Write a PL/SQL code that will reverse a given number like 12345 to 54321.

CODE:

```
DECLARE
 num_to_reverse NUMBER := 12345;
```

```
 reversed_num NUMBER := 0;
 remainder NUMBER;
BEGIN
 WHILE num_to_reverse > 0 LOOP
 remainder := MOD(num_to_reverse, 10);
 reversed_num := (reversed_num * 10) + remainder;
 num_to_reverse := TRUNC(num_to_reverse / 10);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Reversed Number: ' || reversed_num);
END;
```

OUTPUT:

DP2: Write a PL/SQL code to accept emp_id from user and display his / her record. Table: Employee (EmpId, EName, DOJ, DOB, Salary, Address).

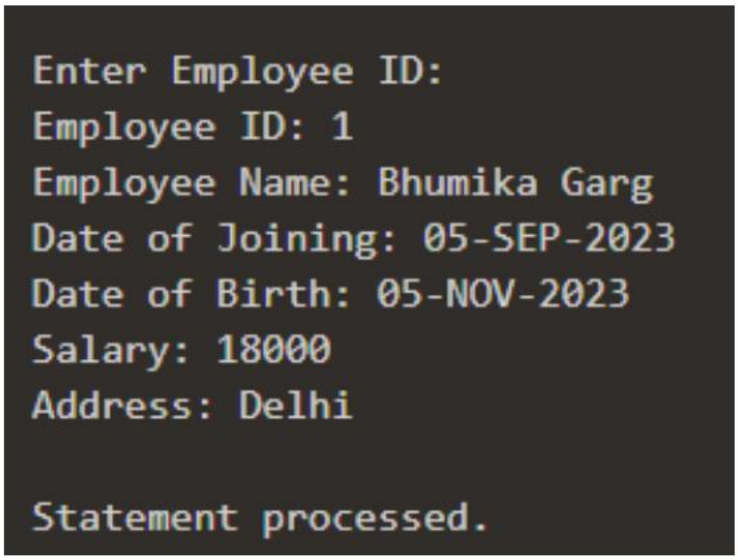
CODE:

```
CREATE TABLE Employee (
 EmpId int PRIMARY KEY,
 EName VARCHAR(50),
 DOJ DATE,
```

```
 DOB DATE,
 Salary int,
 Address VARCHAR(100));
INSERT INTO EMPLOYEE VALUES (1, 'Sidak Ahuja, to_date('2023-09-5', 'YYYY-MM-DD'), to_date('2023-11-5', 'YYYY-MM-DD'), 18000, 'Delhi');
DECLARE
 emp_id_input int;
 emp_name varchar(50);
 emp_doj date;
 emp_dob date;
 emp_salary int;
 emp_address varchar(100);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Enter Employee ID: ');
 emp_id_input :=:Enter_employeed_id;

 SELECT EName, DOJ, DOB, Salary, Address
 INTO emp_name, emp_doj, emp_dob, emp_salary, emp_address
 FROM Employee
 WHERE EmpId = emp_id_input;

 DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_id_input);
 DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
 DBMS_OUTPUT.PUT_LINE('Date of Joining: ' || TO_CHAR(emp_doj, 'DD-MON-YYYY'));
```

```
 DBMS_OUTPUT.PUT_LINE('Date of Birth: ' || TO_CHAR(emp_dob, 'DD-MON-
YYYY'));
 DBMS_OUTPUT.PUT_LINE('Salary: ' || emp_salary);
 DBMS_OUTPUT.PUT_LINE('Address: ' || emp_address);
END;
/
```

OUTPUT:



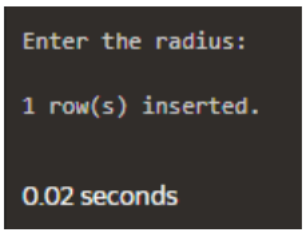
DP3: Write a PL/SQL code to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named Areas, consisting of two columns Radius and Area. Table: Circle (Radius, Area, Circumference).

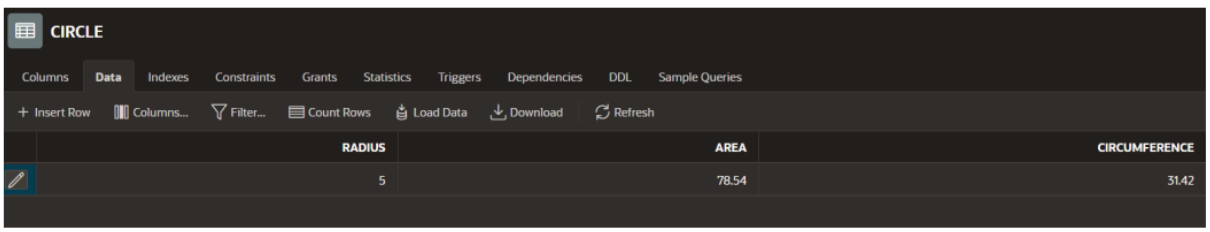
CODE:

```
CREATE TABLE circle (
 Radius int,
 Area decimal(7,2),
 Circumference decimal(7,2)
```

```
);
DECLARE
 radius int;
 c_area decimal(7,2);
 c_circumference decimal(7,2);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Enter the radius: ');
 radius := :radius;
 c_area := round(3.14159 * radius * radius, 2);
 c_circumference := round(3.14159 * 2 * radius, 2);

 INSERT into circle (radius, area, circumference) values (radius, c_area,
c_circumference);
END;
/
```

OUTPUT:





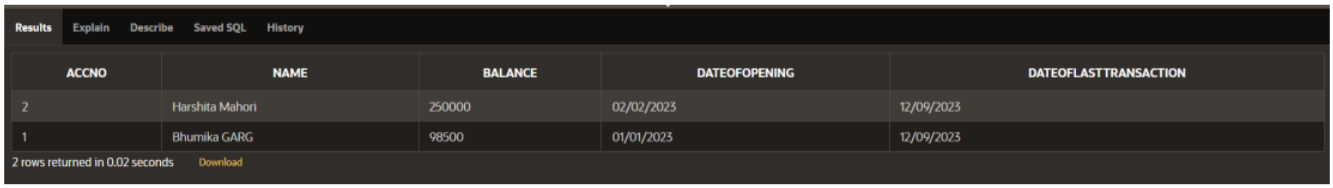
DP4: Write a PL/SQL code that first withdraws an amount of Rs. 1500 from an account. Then deposits an amount of Rs. 50,000 in another account. Update the current balance. Then check to see that the sum of the current balance of all the accounts in the bank does not exceed Rs. 60,00,000. If the balance exceeds Rs 60,00,000 then undo the deposit just made.

CODE:

```
CREATE TABLE Accounts (
 AcCNo INT PRIMARY KEY,
 Name VARCHAR2(50),
 Balance DECIMAL(10,2),
 DateofOpening DATE,
 DateofLastTransaction DATE
);
INSERT INTO Accounts
VALUES (1, 'Bhumika Garg', 100000.00, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-02-01', 'YYYY-MMDD'));
INSERT INTO Accounts
VALUES (2, 'Harshita Mahori', 200000.00, TO_DATE('2023-02-02', 'YYYY-MM-DD'), TO_DATE('2023-02-01', 'YYYYMM-DD'));
DECLARE
 withdraw_amount DECIMAL(10,2) := 1500.00;
 deposit_amount DECIMAL(10,2) := 50000.00;
 total_balance DECIMAL(10,2) := 0.00;
BEGIN
 UPDATE Accounts SET Balance = Balance - withdraw_amount WHERE AcCNo = 1;
 UPDATE Accounts SET Balance = Balance + deposit_amount WHERE AcCNo = 2;
```

```
 UPDATE Accounts SET DateofLastTransaction = SYSDATE WHERE AcCNo IN (1, 2);

 SELECT SUM(Balance) INTO total_balance FROM Accounts;

 IF total_balance > 6000000 THEN

 UPDATE Accounts SET Balance = Balance - deposit_amount WHERE AcCNo = 2;

 DBMS_OUTPUT.PUT_LINE('Deposit undone due to exceeding the total limit of Rs.
60,00,000');

 ELSE

 DBMS_OUTPUT.PUT_LINE('Withdrawal and deposit completed successfully.');

 END IF;

END;

/

select * from accounts;
```

OUTPUT:



Results | Explain | Describe | Saved SQL | History

ACCNO	NAME	BALANCE	DATEOFOPENING	DATEOFLASTTRANSACTION
2	Harshita Mahori	250000	02/02/2023	12/09/2023
1	Bhumika GARG	98500	01/01/2023	12/09/2023

2 rows returned in 0.02 seconds Download

DP5: Write a PL/SQL code that accepts the account number from the user, check the users balance and updates date of last transaction. If it is greater than 1000, only then deduct Rs. 200/- from the balance, otherwise rollback the date of transaction to set it to the previous date. The process is fired on the Acct_Mstr table. Table : Acct_Mstr(AcCNo, Name, Balance, DateofOpening, DateofLastTransaction).

CODE:

```
CREATE TABLE Acct_Mstr (

 AcCNo INT PRIMARY KEY,
```

```
 Name VARCHAR2(50),
 Balance DECIMAL(10,2),
 DateofOpening DATE,
 DateofLastTransaction DATE
);
INSERT INTO Acct_Mstr (AcCNo, Name, Balance, DateofOpening,
DateofLastTransaction)
VALUES (1, 'Sidak Ahuja', 1500.00, TO_DATE('2023-09-05', 'YYYY-MM-DD'),
TO_DATE('2023-01-01', 'YYYY-MMDD'));
DECLARE
 account_number INT;
 current_balance DECIMAL(10,2);
 DateofLastTransaction date;
BEGIN
 account_number := :account_number;
 SELECT Balance, DateofLastTransaction INTO current_balance,
DateofLastTransaction
 FROM Acct_Mstr
 WHERE AcCNo = account_number;
 UPDATE Acct_Mstr
 SET DateofLastTransaction = SYSDATE
 WHERE AcCNo = account_number;
 IF current_balance > 1000 THEN
 UPDATE Acct_Mstr
 SET Balance = Balance - 200
```

```
 WHERE AcCNo = account_number;
 COMMIT;
 ELSE
 UPDATE Acct_Mstr
 SET DateofLastTransaction = DateofLastTransaction
 WHERE AcCNo = account_number;
 ROLLBACK;
 END IF;
END;
/
select * from acct_mstr;
```

ACCNO	NAME	BALANCE	DATEOFOPENING	DATEOFLASTTRANSACTION
1	Bhumika garg	1300	09/05/2023	12/09/2023

1 rows returned in 0.01 seconds Download

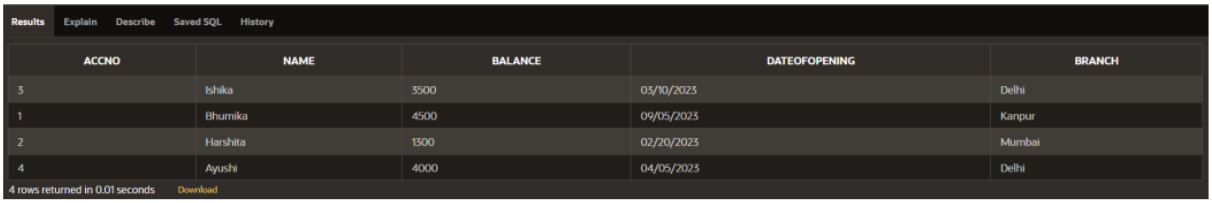
EP:1Write a PL/SQL code that updates the balance of the account holder of 'Delhi' branch and displays the number of records updated. (Use implicit cursor) Table : Accounts(AcCNo, Name, Balance, DateofOpening, Branch).

CODE:

```
CREATE TABLE Accts (
 AcCNo INT PRIMARY KEY,
 Name VARCHAR2(50),
 Balance DECIMAL(10,2),
 DateofOpening DATE,
 Branch VARCHAR2(50)
);
```

```
INSERT INTO Accts VALUES (1, 'Bhumika', 4500.00, TO_DATE('2023-09-05', 'YYYY-MM-DD'), 'Kanpur');

INSERT INTO Accts VALUES (2, 'Harshita', 1300.00, TO_DATE('2023-02-20', 'YYYY-MM-DD'), 'Mumbai');

INSERT INTO Accts VALUES (3, 'Ishika', 3500.00, TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Delhi');

INSERT INTO Accts VALUES (4, 'Ayushi', 4000.00, TO_DATE('2023-04-05', 'YYYY-MM-DD'), 'Delhi');

select * from accts;

DECLARE

 total_updated_records INT := 0;

BEGIN

 FOR account IN (SELECT * FROM Accts WHERE Branch = 'Delhi') LOOP

 UPDATE Accts

 SET Balance = Balance * 1.1 -- increasing by 10%

 WHERE AcCNo = account.AcCNo;

 total_updated_records := total_updated_records + SQL%ROWCOUNT;

 END LOOP;

 DBMS_OUTPUT.PUT_LINE('Number of records updated: ' ||
total_updated_records);

END;

/

select * from accts;
```

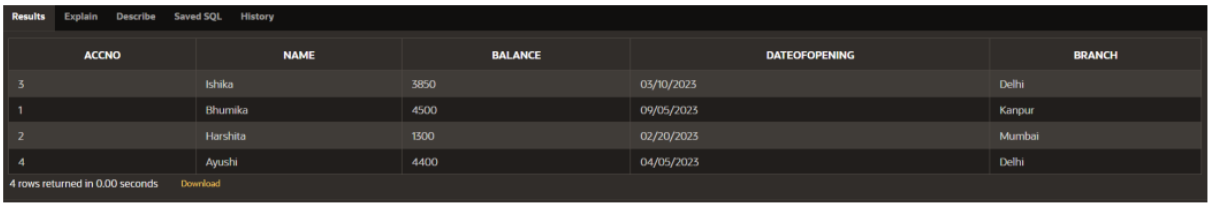
OUTPUT:



Results | Explain | Describe | Saved SQL | History

ACCNO	NAME	BALANCE	DATEOFOPENING	BRANCH
3	Ishika	3500	03/10/2023	Delhi
1	Bhumika	4500	09/05/2023	Kanpur
2	Harshita	1300	02/20/2023	Mumbai
4	Ayushi	4000	04/05/2023	Delhi

4 rows returned in 0.01 seconds Download



Results | Explain | Describe | Saved SQL | History

ACCNO	NAME	BALANCE	DATEOFOPENING	BRANCH
3	Ishika	3850	03/10/2023	Delhi
1	Bhumika	4500	09/05/2023	Kanpur
2	Harshita	1300	02/20/2023	Mumbai
4	Ayushi	4400	04/05/2023	Delhi

4 rows returned in 0.00 seconds Download

EP:2Write a Cursor(PL/SQL code) to display the Employee_name, DateofBirth, Designation whose basic salary is greater than 15000, if not found then show the proper error message. (Use Exception handing) Table : Employee(EmpId, EName, DOJ, DOB, Salary, Address).

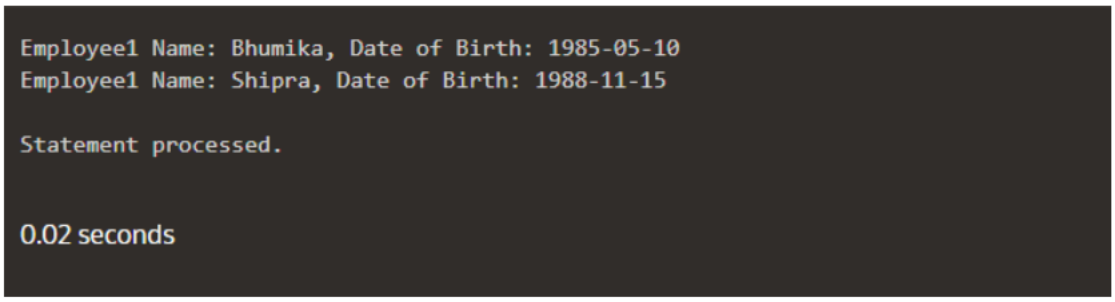
CODE:

```
CREATE TABLE Employee1 (
 EmpId INT PRIMARY KEY,
 EName VARCHAR2(50),
 DOJ DATE,
 DOB DATE,
 Salary DECIMAL(10,2),
 Address VARCHAR2(100)
);
INSERT INTO Employee1 VALUES (12, 'Bhumika', TO_DATE('2018-01-15', 'YYYY-MM-DD'), TO_DATE('1985-05-10',
'YYYY-MM-DD'), 18000.00, 'delhi');
```

```
INSERT INTO Employee1 VALUES (22, 'Harshita', TO_DATE('2019-02-20', 'YYYY-
MM-DD'), TO_DATE('1990-09-22',

'YYYY-MM-DD'), 14000.00, 'Adarsh nagar');

INSERT INTO Employee1 VALUES (32, 'Shipra', TO_DATE('2020-03-10', 'YYYY-MM-
DD'), TO_DATE('1988-11-15', 'YYYYMM-DD'), 17000.00, 'Nagloi');

INSERT INTO Employee1 VALUES (42, 'Ishika', TO_DATE('2021-04-05', 'YYYY-MM-
DD'), TO_DATE('1995-07-28', 'YYYYMM-DD'), 15000.00, ‘kanpur');

DECLARE

 v_EName Employee1.EName%TYPE;

 v_DOB Employee1.DOB%TYPE;

 v_flag BOOLEAN := FALSE;

 SalaryThreshold CONSTANT DECIMAL(10,2) := 15000.00;

 SalaryException EXCEPTION;

BEGIN

 FOR emp IN (SELECT EName, DOB FROM Employee1 WHERE Salary >
SalaryThreshold) LOOP

 v_flag := TRUE;

 v_EName := emp.EName;

 v_DOB := emp.DOB;

 DBMS_OUTPUT.PUT_LINE('Employee1 Name: ' || v_EName || ', Date of Birth: '
|| TO_CHAR(v_DOB, 'YYYY-MMDD'));

 END LOOP;

 IF v_flag = FALSE THEN

 RAISE SalaryException;

 END IF;

 EXCEPTION
```

```
 WHEN SalaryException THEN
 DBMS_OUTPUT.PUT_LINE('No employees found with salary greater than ' ||
SalaryThreshold);
END;
/
```

OUTPUT:



FP1: Create a procedure that accepts the emp id from the calling procedure and displays his/her record. Table : Employee(EmpId, EName, DOJ, DOB, Salary, Address).

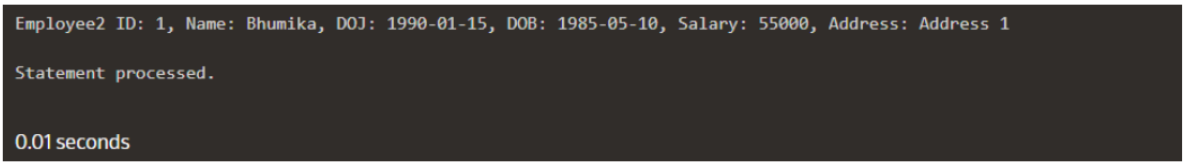
CODE:

```
CREATE TABLE Employee2 (
 EmpId INT PRIMARY KEY,
 EName VARCHAR2(50),
 DOJ DATE,
 DOB DATE,
 Salary DECIMAL(10,2),
 Address VARCHAR2(100)
);
```

```
INSERT INTO Employee2 VALUES (1, 'Bhumika', TO_DATE('1990-01-15', 'YYYY-MM-DD'), TO_DATE('1985-05-10',

'YYYY-MM-DD'), 55000.00, 'Address 1');

INSERT INTO Employee2 VALUES (2, 'Harshita', TO_DATE('1991-02-20', 'YYYY-MM-DD'), TO_DATE('1990-09-22',

'YYYY-MM-DD'), 45000.00, 'Address 2');

INSERT INTO Employee2 VALUES (3, 'Ishika', TO_DATE('1992-03-10', 'YYYY-MM-DD'), TO_DATE('1988-11-15', 'YYYYMM-DD'), 35000.00, 'Address 3');

INSERT INTO Employee2 VALUES (4, 'Ayushi', TO_DATE('1993-04-05', 'YYYY-MM-DD'), TO_DATE('1995-07-28', 'YYYYMM-DD'), 25000.00, 'Address 4');

CREATE OR REPLACE PROCEDURE DisplayEmployeeDetails(

 emp_id_in IN Employee2.EmpId%TYPE

)

IS

 v_EName Employee2.EName%TYPE;

 v_DOJ Employee2.DOJ%TYPE;

 v_DOB Employee2.DOB%TYPE;

 v_Salary Employee2.Salary%TYPE;

 v_Address Employee2.Address%TYPE;

BEGIN

 SELECT EName, DOJ, DOB, Salary, Address

 INTO v_EName, v_DOJ, v_DOB, v_Salary, v_Address

 FROM Employee2

 WHERE EmpId = emp_id_in;

 DBMS_OUTPUT.PUT_LINE('Employee2 ID: ' || emp_id_in || ', Name: ' || v_EName || ', DOJ: ' || TO_CHAR(v_DOJ,
```

```
'YYYY-MM-DD')
 || ', DOB: ' || TO_CHAR(v_DOB, 'YYYY-MM-DD') || ', Salary: ' || v_Salary || ', Address: ' || v_Address);
END;
/
BEGIN
 DisplayEmployeeDetails(1);
END;
/
```

OUTPUT:



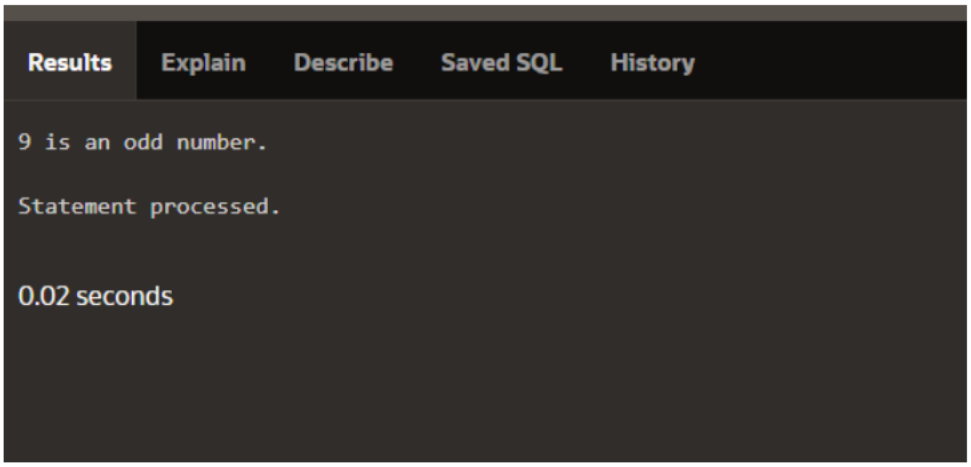
FP:2 Write a PL/SQL function ODDEVEN to return value TRUE if the number passed to it is EVEN else will return FALSE.

CODE:

```
CREATE OR REPLACE FUNCTION ODDEVEN(
 num IN NUMBER
) RETURN BOOLEAN
IS
BEGIN
 IF MOD(num, 2) = 0 THEN
 RETURN TRUE;
 ELSE
```

```
 RETURN FALSE;
 END IF;
END;
/
DECLARE
 input_number NUMBER := :num_inp;
 result BOOLEAN;
BEGIN
 result := ODDEVEN(input_number);
 IF result THEN
 DBMS_OUTPUT.PUT_LINE(input_number || ' is an even number.');
 ELSE
 DBMS_OUTPUT.PUT_LINE(input_number || ' is an odd number.');
 END IF;
END;
/
```

OUTPUT:


Results
Explain
Describe
Saved SQL
History
9 is an odd number.
Statement processed.
0.02 seconds