Group A
Assignment Number 2 (a)
PL/SQL

Problem Statement:

Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym.

PROGRAM INPUT

---

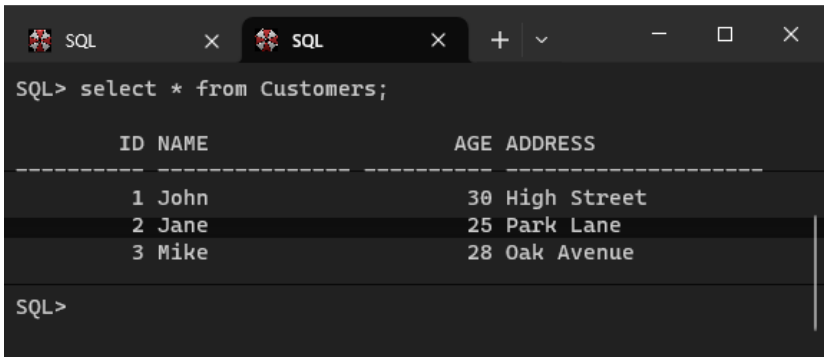
1. Table

```
CREATE TABLE Customers (
   ID int PRIMARY KEY,
   Name varchar(15),
   Age int,
   Address varchar(20)
   );


INSERT INTO Customers VALUES (1, 'John', 30, 'High Street');

INSERT INTO Customers VALUES (2, 'Jane', 25, 'Park Lane');

INSERT INTO Customers VALUES (3, 'Mike', 28, 'Oak Avenue');
```



2. View

```
CREATE VIEW CustomersOver30 AS
  2  SELECT * FROM Customers
  3  WHERE Age>=30;
```

```
SQL> select * from CustomersOver30;

        ID NAME                           AGE ADDRESS
---------- --------------- ---------- --------------------
         1 John                            30 High Street

SQL>
```

3. Index

CREATE INDEX idx_age ON Customers (Age);

4. Sequence

CREATE SEQUENCE customer_id_seq;

```
SQL> select customer_id_seq.nextval
  2  from dual;

   NEXTVAL
----------
         1

SQL>
```

5. Synonym

CREATE SYNONYM customer_table FOR Customers;

```
SQL> select * from customer_table;

        ID NAME                           AGE ADDRESS
---------- --------------- ---------- --------------------
         1 John                            30 High Street
         2 Jane                            25 Park Lane
         3 Mike                            28 Oak Avenue

SQL>
```

Group A
Assignment Number 2 (b)
PL/SQL

Problem Statement:

Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, delete with operators, functions and set operator.

PROGRAM INPUT

```
CREATE TABLE Products (
   ProductID INT PRIMARY KEY,
   ProductName VARCHAR(255) NOT NULL,
   CategoryID INT,
   Price DECIMAL(10, 2) NOT NULL
);


CREATE TABLE Categories (
   CategoryID INT PRIMARY KEY,
   CategoryName VARCHAR(50) NOT NULL
);


INSERT INTO Categories (CategoryID, CategoryName)
VALUES
   (1, 'Electronics'),
   (2, 'Clothing'),
   (3, 'Kitchen');


INSERT INTO Products (ProductID, ProductName, CategoryID, Price)
VALUES
   (1, 'Smartphone', 1, 599.99),
   (2, 'Laptop', 1, 1299.99),
   (3, 'T-Shirt', 2, 19.99),
   (4, 'Dress', 2, 49.99),
   (5, 'Coffee Maker', 3, 79.99);
```

```
mysql> select * from Products;
+-----------+--------------+------------+---------+
| ProductID | ProductName  | CategoryID | Price   |
+-----------+--------------+------------+---------+
|         1 | Smartphone   |          1 |  599.99 |
|         2 | Laptop       |          1 | 1299.99 |
|         3 | T-Shirt      |          2 |   19.99 |
|         4 | Dress        |          2 |   49.99 |
|         5 | Coffee Maker |          3 |   79.99 |
+-----------+--------------+------------+---------+
5 rows in set (0.00 sec)
```

```
mysql> select * from Categories;
+------------+--------------+
| CategoryID | CategoryName |
+------------+--------------+
|          1 | Electronics  |
|          2 | Clothing     |
|          3 | Kitchen      |
+------------+--------------+
3 rows in set (0.00 sec)

mysql>
```

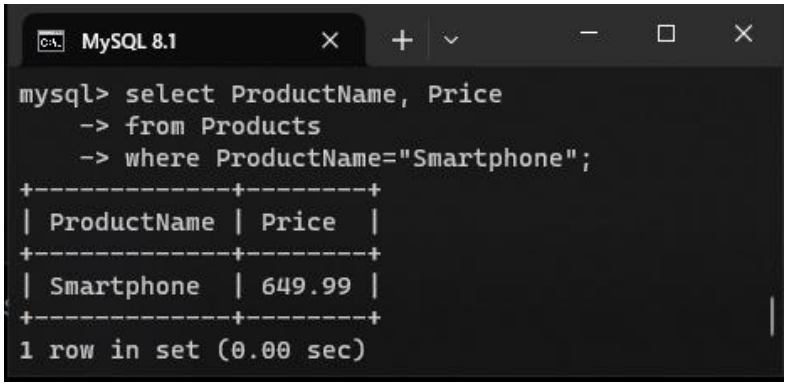
1. **Select all products in the "Electronics" category**

SELECT ProductName, Price
FROM Products
WHERE CategoryID = 1;

```
+-------------+---------+
| ProductName | Price   |
+-------------+---------+
| Smartphone  |  599.99 |
| Laptop      | 1299.99 |
+-------------+---------+
```

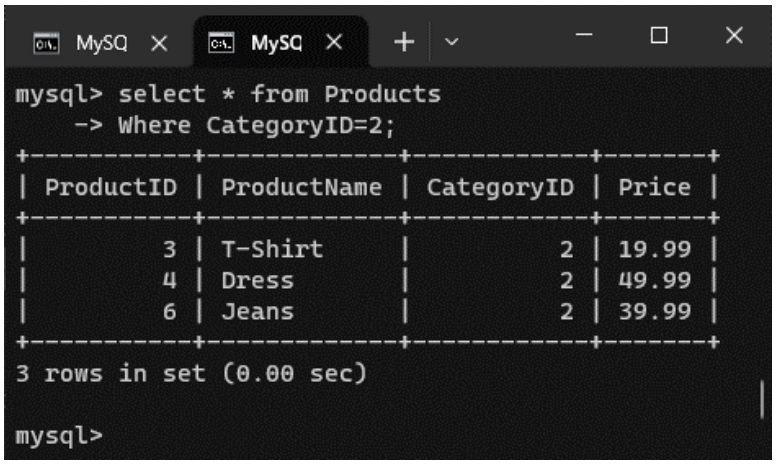
2. **Update the price of the "Smartphone"**

UPDATE Products
SET Price = 649.99
WHERE ProductName = 'Smartphone';



3. **Insert a new product into the "Clothing" category**

INSERT INTO Products (ProductID, ProductName, CategoryID, Price)
VALUES (6, 'Jeans', 2, 39.99);



4. **Select products with a price less than $50**

SELECT ProductName, Price
FROM Products
WHERE Price < 50.00;

```
MySQL 8.1
+-------------+-------+
| ProductName | Price |
+-------------+-------+
| T-Shirt     | 19.99 |
| Jeans       | 39.99 |
+-------------+-------+
2 rows in set (0.00 sec)

mysql>
```

5. **Delete the “Dress” product**

DELETE FROM Products
WHERE ProductName = 'Dress';

```
MySQL 8.1
mysql> select * from Products;
+-----------+--------------+------------+---------+
| ProductID | ProductName  | CategoryID | Price   |
+-----------+--------------+------------+---------+
|         1 | Smartphone   |          1 |  649.99 |
|         2 | Laptop       |          1 | 1299.99 |
|         3 | T-Shirt      |          2 |   19.99 |
|         5 | Coffee Maker |          3 |   79.99 |
|         6 | Jeans        |          2 |   39.99 |
+-----------+--------------+------------+---------+
```

6. **Calculate the average price of products in the “Electronics” category**

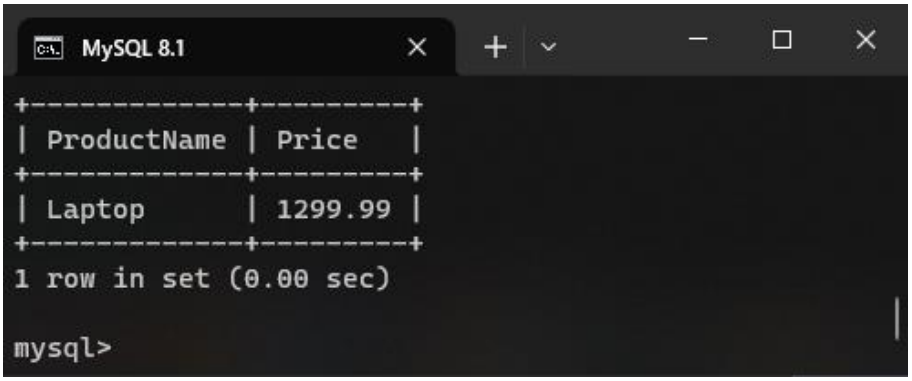
SELECT AVG(Price)
FROM Products
WHERE CategoryID = 1;

```
MySQL 8.1
+------------+
| AVG(Price) |
+------------+
| 974.990000 |
+------------+
1 row in set (0.00 sec)

mysql>
```

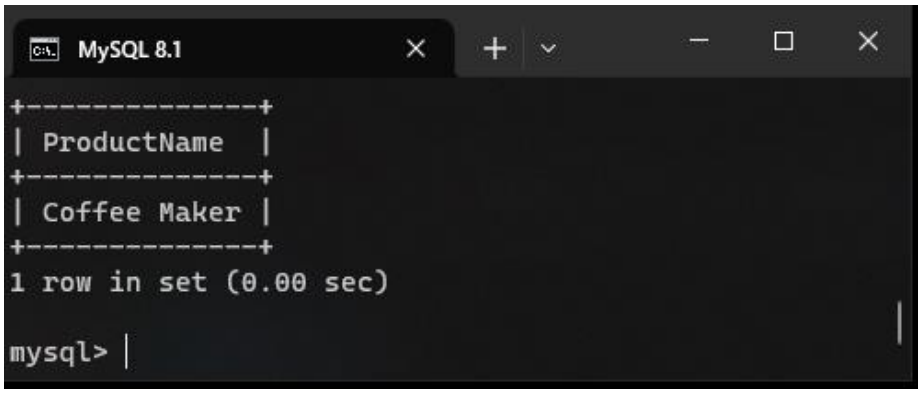
7. **Select the most expensive product**

```
SELECT ProductName, Price
FROM Products
WHERE Price = (SELECT MAX(Price) FROM Products);
```



8. **Find products that contain the word "Coffee" in their name**

```
SELECT ProductName
FROM Products
WHERE ProductName LIKE '%Coffee%';
```



9. **Retrieve the total number of products in each category**

```
SELECT Categories.CategoryName, COUNT(Products.ProductID) AS TotalProducts
FROM Categories
LEFT JOIN Products ON Categories.CategoryID = Products.CategoryID
GROUP BY Categories.CategoryName;
```

```
MySQL 8.1
+--------------+---------------+
| Electronics  |             2 |
| Clothing     |             2 |
| Kitchen      |             1 |
+--------------+---------------+
3 rows in set (0.00 sec)

mysql>
```

10. Find all products with prices greater than the average price of products

```
SELECT ProductName, Price
FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);
```

```
MySQL 8.1
+-------------+---------+
| ProductName | Price   |
+-------------+---------+
| Smartphone  |  649.99 |
| Laptop      | 1299.99 |
+-------------+---------+
2 rows in set (0.00 sec)
```

Group A
Assignment Number 3
PL/SQL

Problem Statement:

SQL Queries – all types of Join, Sub-Query and View: Write at least10 SQL queries for suitable database application using SQL DML statements.

PROGRAM INPUT

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Email VARCHAR(100),
   Country VARCHAR(50)
);


INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Country)
VALUES
   (1, 'John', 'Doe', 'john.doe@email.com', 'USA'),
   (2, 'Alice', 'Smith', 'alice.smith@email.com', 'Canada'),
   (3, 'Bob', 'Johnson', 'bob.johnson@email.com', 'UK');


CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   CustomerID INT,
   OrderDate DATE,
   TotalAmount DECIMAL(10, 2)
);


INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES
   (101, 1, '2023-01-15', 100.00),
   (102, 2, '2023-01-16', 75.50),
   (103, 3, '2023-01-16', 200.25);
```

```
mysql> select * from Customers;
+------------+-----------+----------+-----------------------+---------+
| CustomerID | FirstName | LastName | Email                 | Country |
+------------+-----------+----------+-----------------------+---------+
|          1 | John      | Doe      | john.doe@email.com    | USA     |
|          2 | Alice     | Smith    | alice.smith@email.com | Canada  |
|          3 | Bob       | Johnson  | bob.johnson@email.com | UK      |
+------------+-----------+----------+-----------------------+---------+
3 rows in set (0.00 sec)
```

```
mysql> select * from orders;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|     101 |          1 | 2023-01-15 |      100.00 |
|     102 |          2 | 2023-01-16 |       75.50 |
|     103 |          3 | 2023-01-16 |      200.25 |
+---------+------------+------------+-------------+
3 rows in set (0.00 sec)

mysql>
```

1. Inner join to retrieve customer names and order dates

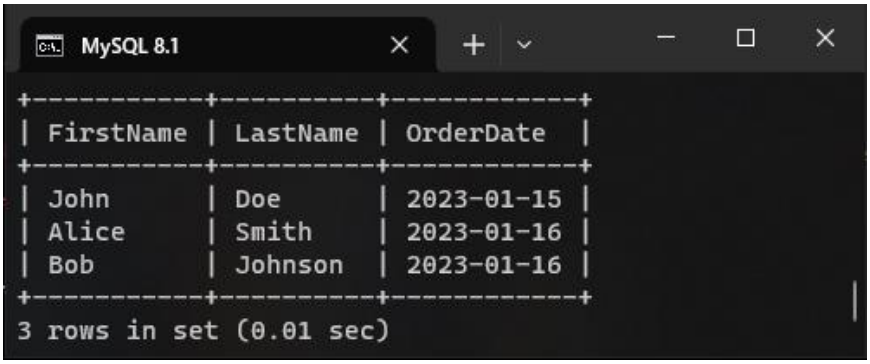
SELECT c.FirstName, c.LastName, o.OrderDate
FROM Customers c
INNER JOIN Orders o ON c.CustomerID = o.CustomerID;

```
+-----------+----------+------------+
| FirstName | LastName | OrderDate  |
+-----------+----------+------------+
| John      | Doe      | 2023-01-15 |
| Alice     | Smith    | 2023-01-16 |
| Bob       | Johnson  | 2023-01-16 |
+-----------+----------+------------+
3 rows in set (0.00 sec)

mysql>
```

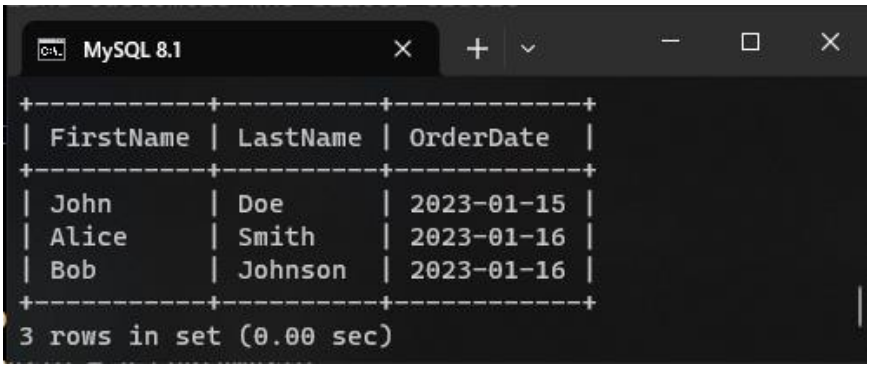
2. Left Join to retrieve all customers and their orders (even if they haven't placed any orders)

```
SELECT c.FirstName, c.LastName, o.OrderDate
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;
```



3. Right Join to retrieve all orders and the customer information (even if customer details are missing)

```
SELECT c.FirstName, c.LastName, o.OrderDate
FROM Customers c
RIGHT JOIN Orders o ON c.CustomerID = o.CustomerID;
```

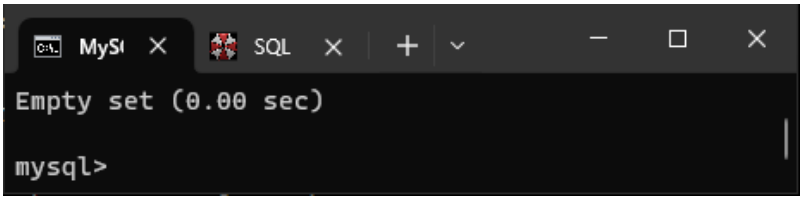


4. Full Outer Join to retrieve all customers and orders (combining results from Left and Right Joins)

```
SELECT c.FirstName, c.LastName, o.OrderDate
FROM Customers c
FULL OUTER JOIN Orders o ON c.CustomerID = o.CustomerID;
```

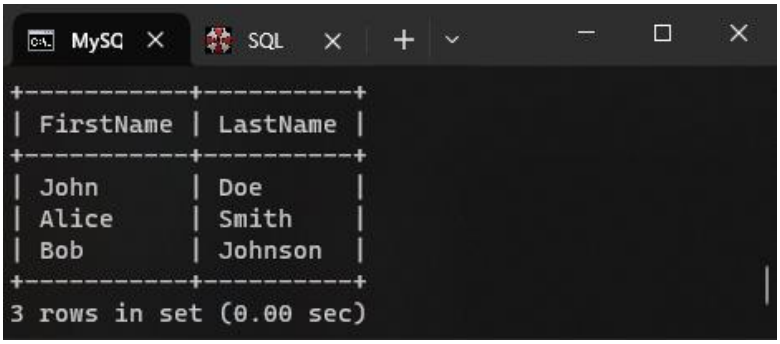
5. Self-Join to find customers from the same country

```
SELECT c1.FirstName, c1.LastName, c2.FirstName AS FriendFirstName, c2.LastName AS FriendLastName
FROM Customers c1
JOIN Customers c2 ON c1.Country = c2.Country AND c1.CustomerID <> c2.CustomerID;
```



6. Subquery to find customers who placed orders

```
SELECT FirstName, LastName
FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Orders);
```

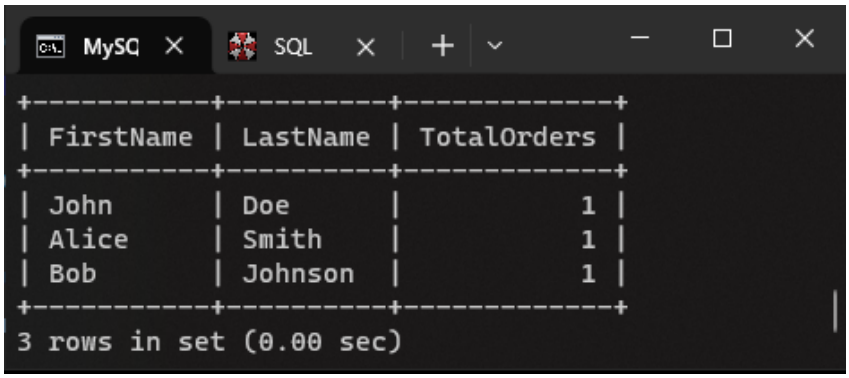


FirstName	LastName
John	Doe
Alice	Smith
Bob	Johnson

3 rows in set (0.00 sec)

7. Subquery to find the total number of orders placed by each customer

```
SELECT FirstName, LastName, (
   SELECT COUNT(OrderID) FROM Orders o
   WHERE o.CustomerID = c.CustomerID
) AS TotalOrders
FROM Customers c;
```

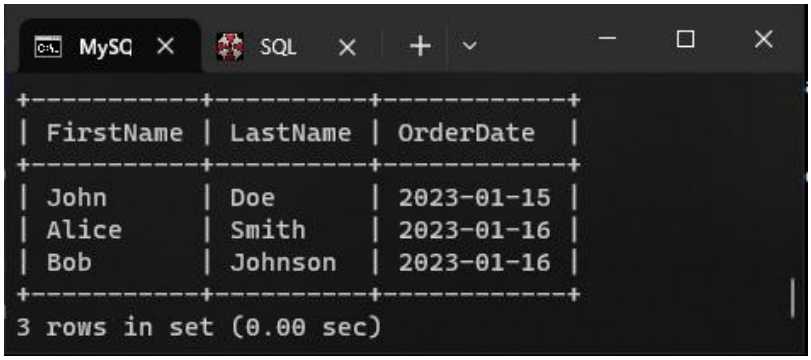


FirstName	LastName	TotalOrders
John	Doe	1
Alice	Smith	1
Bob	Johnson	1

3 rows in set (0.00 sec)

8. View to simplify query complexity

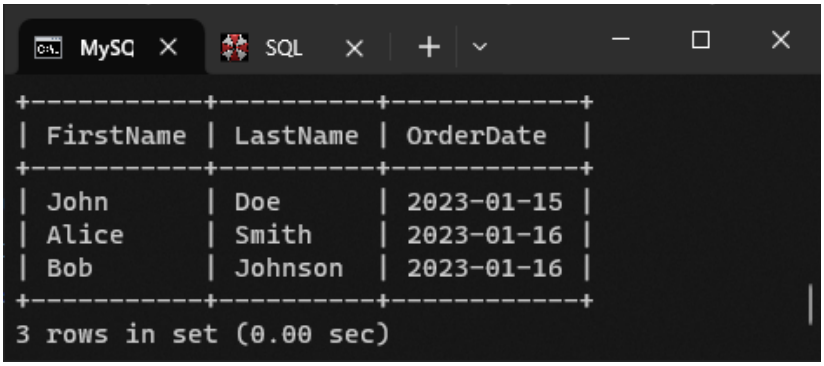
```
CREATE VIEW CustomerOrders AS
SELECT c.FirstName, c.LastName, o.OrderDate
FROM Customers c
INNER JOIN Orders o ON c.CustomerID = o.CustomerID;
```



```
+-----------+----------+------------+
| FirstName | LastName | OrderDate  |
+-----------+----------+------------+
| John      | Doe      | 2023-01-15 |
| Alice     | Smith    | 2023-01-16 |
| Bob       | Johnson  | 2023-01-16 |
+-----------+----------+------------+
3 rows in set (0.00 sec)
```

9. Using a View to retrieve customer names and order dates

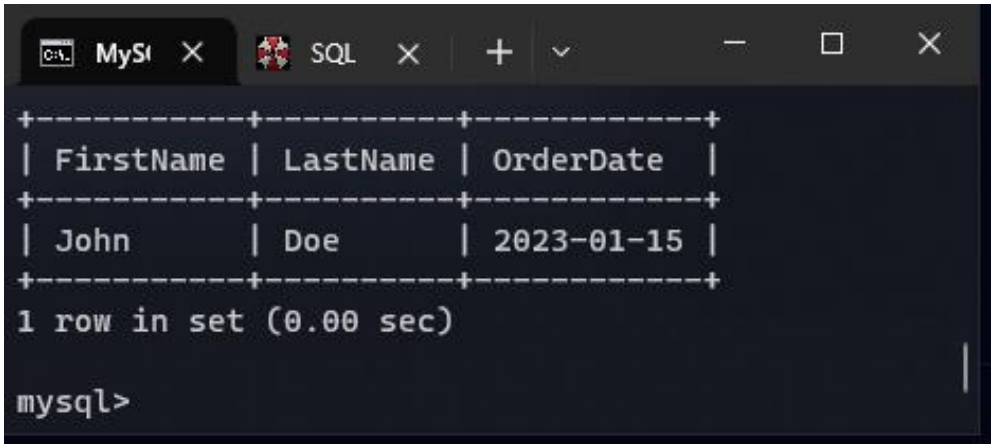
```
SELECT FirstName, LastName, OrderDate
FROM CustomerOrders;
```



```
+-----------+----------+------------+
| FirstName | LastName | OrderDate  |
+-----------+----------+------------+
| John      | Doe      | 2023-01-15 |
| Alice     | Smith    | 2023-01-16 |
| Bob       | Johnson  | 2023-01-16 |
+-----------+----------+------------+
3 rows in set (0.00 sec)
```

10. Using a View to find customers who placed orders in a specific country

```
SELECT co.FirstName, co.LastName, co.OrderDate
FROM CustomerOrders co
Inner Join Customers c on c.FirstName = co.FirstName
WHERE c.Country = 'USA';
```



```
+-----------+----------+------------+
| FirstName | LastName | OrderDate  |
+-----------+----------+------------+
| John      | Doe      | 2023-01-15 |
+-----------+----------+------------+
1 row in set (0.00 sec)

mysql>
```

Group A
Assignment Number 4
PL/SQL

Problem Statement:

Unnamed PL/SQL code block: Use of Control structure and Excepon handling is mandatory.
Schema:
1. Borrower(Roll_no, Name, DateofIssue, NameofBook, Status) 2.
Fine(Roll no,Date, Amt) Accept roll no & name of book from user.
Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs Sper day.
If no, of days 30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. Aer subming the book, status will change from I to R.
If condion of fine is true, then details will be stored into fine table.

PROGRAM INPUT

```
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 (54, ‘SUDARSHAN’, TO_DATE ('01-10-2022, DD-MM-YYYY'), 'HARRY
POTTER','I’);

INSERT INTO borrower VALUES(56,'SUMIT’, TO_DATE(‘15-10-2022','DD-MM-YYYY'),  ’DARK
MATTER','I');

INSERT INTO borrower VALUES(68, 'MANDAR’, TO_DATE(‘24-09-2022','DD-MM- YYYY'),
 'SILENT HILL','I');

INSERT INTO borrower VALUES(66, 'SIDDHAM”, TO_DATE(26-08-2022','DD-MM- YYYY'), 'GOD
OF WAR','I’);

INSERT INTO borrower VALUES(50,'SHREYAS',TO_DATE(09-09-2022','DD-MM-YYYY'), 'SPIDER-MAN','I');
```

```
select *from borrower
```

Results Explain Describe Saved SQL History

ROLL_NO	NAME	DATEOFISSUE	NAME_OF_BOOK	STATUS
54	SUDARSHAN	10/01/2022	HARRY POTTER	I
56	SUMIT	10/15/2022	DARK MATTER	I
68	MANDAR	09/24/2022	SILENT HILL	I
66	SOHAM	08/26/2022	GOD OF WAR	I
50	SHREYAS	09/09/2022	SPIDER-MAN	I

5 rows returned in 0.02 seconds Download

```
DECLARE i_roll_no NUMBER: 54; 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: '&nameoook';  --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 = 'R'
        WHERE borrower.roll_no=i_roll_no;

END;
```

FINE TABLE AFTER SUBMITTING :

Autocommit Rows 10 Save Run

```
select * from fine
```

Results Explain Describe Saved SQL History

ROLL_NO	DATE_OF_RETURN	AMT
54	10/18/2023	30984.7887731481481481481481481481481482
54	10/18/2023	30984.7893518518518518518518518518518519

2 rows returned in 0.00 seconds Download

BORROWER TABLE AFTER SUBMITTING :

Autocommit Rows 10 Save Run

```
select * from borrower
```

Results Explain Describe Saved SQL History

ROLL_NO	NAME	DATEOFISSUE	NAME_OF_BOOK	STATUS
54	SUDARSHAN	10/01/2022	HARRY POTTER	R
56	SUMIT	10/15/2022	DARK MATTER	I
68	MANDAR	09/24/2022	SILENT HILL	I
66	SOHAM	08/26/2022	GOD OF WAR	I
50	SHREYAS	09/09/2022	SPIDER-MAN	R

5 rows returned in 0.00 seconds Download

Assignment Number 5
PL/SQL

Problem Statement:
Write a a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to . Store the radius and the corresponding values of calculated area in an empty table named areas, consisng of two columns, radius and area.

Lab Exercise:

1. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 6 to 10. Store the radius and the corresponding values of calculated area in an empty table named areas, consisng of two columns, radius and area.
2. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 10 to 15. Store the radius and the corresponding values of calculated area in an empty table named areas, consisng of two columns, radius and area.
3. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 4 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisng of two columns, radius and area.

PROGRAM INPUT & OUTPUT

```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2)
);

DECLARE pi constant number(4,2) :=
        3.14;
        area number(14,2); radius
        number(5);

BEGIN
        Radius := 5; while radius <= 9 loop
        area := pi*power(radius,2); insert
        into areas values(radius,area);
        radius := radius+1; end loop; end;

SELECT *FROM AREAS;
```

Autocommit Rows 10 Save Run

```
select *from ar;
```

Results Explain Describe Saved SQL History

RADIUS	AREA
5	78.5
6	113.04
7	153.86
8	200.96
9	254.34

5 rows returned in 0.00 seconds Download

```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2)
);

DECLARE
        pi constant number(4,2) := 3.14;
        area number(14,2);
        radius number(5);

BEGIN
        Radius := 6; while radius <= 10 loop
        area := pi*power(radius,2); insert
        into areas values(radius,area);
        radius := radius+1; end loop;
        end;

SELECT *FROM AREAS
```

Autocommit Rows 10 Save Run

```
SELECT *FROM A
```

Results Explain Describe Saved SQL History

RADIUS	AREA
6	113.04
7	153.86
8	200.96
9	254.34
10	314

5 rows returned in 0.00 seconds Download

```
CREATE TABLE AREAS
(
RADIUS NUMBER(5),
AREA NUMBER(14,2)
);

DECLARE
        pi constant number(4,2) := 3.14;
        area number(14,2);
        radius number(5);

BEGIN
        Radius := 10; while radius <= 15 loop
        area := pi*power(radius,2); insert
        into areas values(radius,area);
        radius := radius+1; end loop;
        end;

SELECT *FROM AREAS
```

Autocommit Rows 10 Save Run

```
SELECT *FROM Ae
```

Results Explain Describe Saved SQL History

RADIUS	AREA
10	314
11	379.94
12	452.16
13	530.66
14	615.44
15	706.5

6 rows returned in 0.00 seconds Download

Assignment Number 6
PL/SQL

Problem Statement:
Named PL/SQL Block: PL/SQL Stored Procedure and Stored Funcon.
Write a Stored Procedure namely proc Grade for the categorizaon of student. If marks scored by students in examinaon is 1500 and marks-990then student will be placed in disncon category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQL block for using procedure created with above requirement. Stud Marks(name, total marks) Result(Roll Name, Class).

PROGRAM INPUT

```
CREATE TABLE stud marks
(
name VARCHAR2(25),
total_marks NUMBER
);

CREATE TABLE result
(
roll_number NUMBER, name
VARCHAR2(25),
class VARCHAR2(30)
);

CREATE OR REPLACE PROCEDURE procedure 1 (roll no IN NUMBER, name IN VARCHAR2 marks IN
NUMBER)
AS
BEGIN
        IF (marks<=1500 and marks>=990) THEN
                DBMS_OUTPUT.PUT LINE (roll_no||" - "||name||’ : DISTINCTION");
                INSERT INTO result VALUES (roll_no,name, ‘DISTINCTION’);
        ELSIF (marks<=989 and marks>=900) THEN
                DBMS_OUTPUT.PUT_LINE (roll_no||’-‘||name||’ : FIRST CLASS');
                INSERT INTO result VALUES (roll no,name, 'FIRST CLASS');
        ELSIF (marks<=899 and marks>825) THEN
                DBMS_OUTPUT.PUT_LINE(roll_no||"-"||name||’: HIGHER SECOND CLASS'); INSERT
                INTO result VALUES (roll_no,name, 'HIGHER SECOND CLASS');
        ELSE
                DBMS_OUTPUT.PUT_LINE (roll_no||’-‘||name||’: FAIL');
                INSERT INTO result VALUES (roll_no,name, 'FAIL'); END IF;
        END IF;
                INSERT INTO stud_marks VALUES (name,marks);
END procedure_1;
```

```
BEGIN
        Procedure_1(54,’ SUDARSHAN’,1000);  Procedure_1(46,'ARYAN’,
        1950).
        Procedure_1(58,'ARJUN’, 1050);
        Procedure_1(48,'SARTHAK ',750);
END;
```

OUTPUT:

Autocommit Rows 10 Save Run

```
BEGIN
  procedure_1(54, 'SUDARSHAN', 1000);
  procedure_1(46, 'ARYAN', 1950);
  procedure_1(58, 'ARJUN', 1050);
  procedure_1(48, 'SARTHAK', 750);
END;
```

Results Explain Describe Saved SQL History

```
54 - SUDARSHAN : DISTINCTION
46 - ARYAN : FAIL
58 - ARJUN : DISTINCTION
48 - SARTHAK : FAIL

Statement processed.
```

Autocommit Rows 10 Save Run

```
select * from resul;
```

Results Explain Describe Saved SQL History

ROLL_NUMBER	NAME	CLASS
54	SUDARSHAN	DISTINCTION
46	ARYAN	FAIL
58	ARJUN	DISTINCTION
48	SARTHAK	FAIL

4 rows returned in 0.00 seconds Download

Autocommit Rows 10 Save Run

```
select * from stud;
```

Results Explain Describe Saved SQL History

NAME	TOTAL_MARKS
SUDARSHAN	1000
ARYAN	1950
ARJUN	1050
SARTHAK	750

4 rows returned in 0.00 seconds Download

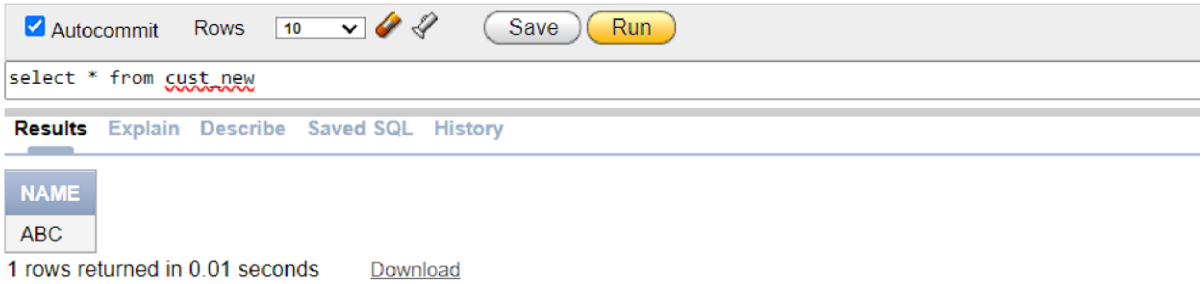
Assignment Number 7
Cursors

Problem Statement:
cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)
Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table Cust New with the data available in the table Cust Old. If the data in the first table already exist in the second table then that data should be skipped.

PROGRAM INPUT & OUTPUT

```
CREATE TABLE Cust _New
(
Name VARCHAR2(15)
);

INSERT INTO Cust_New VALUES (“ABC");
```




```
CREATE TABLE Cust_Old
(
Name VARCHAR2(15)
);

INSERT INTO Cust Old VALUES (‘ABC’);
INSERT INTO Cust Old VALUES (‘PQR’);
INSERT INTO Cust Old VALUES (‘XYZ’);
```

Autocommit Rows 10 Save Run

```
select * from cust_old;
```

Results Explain Describe Saved SQL History

NAME
XYZ
ABC
PQR

3 rows returned in 0.00 seconds Download

```
DECLARE

CURSOR cur1 IS
SELECT Name from Cust_Old;

CURSOR cur2 IS
SELECT Name from Cust_New;

R VARCHAR(15);
C_Name  VARCHAR(15);

BEGIN
OPEN cur1;
OPEN cur2;

LOOP
Fetch cur1 into C_Name;
Fetch cur2 into R;

EXIT WHEN cur1%FOUND = FALSE;  IF
 R <> C_Name THEN
INSERT INTO Cust_New VALUES (C_Name);
END IF;
END LOOP;
CLOSE cur1;
END;

SELECT * FROM Cust_New;
```

Autocommit Rows 10 Save Run

```
SELECT * FROM Cust_New;
```

Results Explain Describe Saved SQL History

NAME
ABC
XYZ
PQR

3 rows returned in 0.00 seconds Download

Assignment Number 8

Trigger

Problem Statement:

Database Trigger (All Types: Row level and Statement level triggers, Before and Aer Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library Audit table.

PROGRAM INPUT

---Table Creaon

```
CREATE TABLE Library
(
Book_Id NUMBER(5),
Book_Name VARCHAR2(20),
Book_Type VARCHAR2(20),  Issued_By
VARCHAR2(20)
):
```

:**Table Inseron**

```
INSERT INTO Library VALUES (1234, ‘DBMS', 'Reference','Sudarshan');
INSERT INTO Library VALUES (1836, ‘TOC’, ‘Text’, Siddham');
INSERT INTO Library VALUES (1996, ‘SPOS', 'Reference', 'Shreyas');
INSERT INTO Library VALUES (1196, ‘CNS', ‘Text’, ‘Sairaj');
```

Autocommit Rows 10 Save Run

```
select * from library
```

Results Explain Describe Saved SQL History

BOOK_ID	BOOK_NAME	BOOK_TYPE	ISSUED_BY
1234	DBMS	Reference	Sudarshan
1996	SPOS	Reference	Shreyas
1196	CNS	Text	Sairaj
1836	TOC	Text	Siddham

4 rows returned in 0.00 seconds Download

;Table Creaon

```
CREATE TABLE Back_UP
(
 Book_Id NUMBER(5),
Book_Name VARCHAR2(20),
Book_Type VARCHAR2(20),
Issued_By VARCHAR2(20)
);
```

: Trigger Creaon

```
CREATE TRIGGER Update_Rec
AFTER UPDATE OR DELETE ON Library
FOR EACH ROW

BEGIN
        INSERT INTO Back_UP (Book_Id, Book_Name, Book Type, Issued By) VALUES
          (cold.Book_Id, :old.Book_Name, :old.Book_Type, :old.Issued_By);
END;

UPDATE LIBRARY
SET Issued By = 'Sairaj'
WHERE Issued By = 'Sumit';

 SELECT FROM Back_UP;
```

Assignment 2(a) – DDL

1. Trains table created

```
create table train
(
train_id number,
train_name varchar(50),
departure_station varchar(50),
arrival_station varchar(50),
departure_time timestamp,
arrival_time timestamp
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

2. Staon table created

Autocommit Rows 10 Save Run

```
create table station
(
station_id number,
station_name varchar(50),
loacation varchar(50),
platform int
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.03 seconds

3. Passenger table created

Autocommit Rows 10 Save Run

```
create table passenger
(
passenger_id number,
name varchar(50),
gender varchar(50),
age int,
contact_no number,
email varchar(100)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.01 seconds

4. Ticket prices table created

Autocommit Rows 10 Save Run

```
create table ticket_prices
(
ticket_id number,
train_id number,
class varchar(50),
price number
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.02 seconds

5. Reservaon table created

Autocommit Rows 10 Save Run

```
create table reservation
(
reservation_id number,
train_id number,
passenger_id number,
booking_date date,
seat_number number,
status varchar(100)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.01 seconds

6. User table created

Autocommit Rows 10 Save Run

```
create table users
(
user_id number,
user_name varchar(100),
password varchar(8),
role varchar(50)
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.00 seconds

7. Trains table altered

Autocommit Rows 10 Save Run

```
alter table train
add primary key (train_id);
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.05 seconds

8. Staon table altered

```
alter table station
add primary key (station_id);
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.02 seconds

9. Passenger table altered

Autocommit Rows 10 Save Run

```
alter table passenger
add primary key (passenger_id);
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.01 seconds

10. Ticket prices table altered

Autocommit Rows 10 Save Run

```
alter table ticket_prices
add primary key (ticket_id);
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.01 seconds

11. Reservaon table altered

Autocommit Rows 10 Save Run

```
alter table reservation
add primary key (reservation_id);
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.01 seconds

12. User table altered

```
alter table users
add primary key (user_id);
```

Results Explain Describe Saved SQL History

```
Table altered.
```

13. Views in SQL

Autocommit Rows 10 Save Run

```
create view train_view as
select * from train
```

Results Explain Describe Saved SQL History

```
View created.
```

0.03 seconds

Autocommit Rows 10 Save Run

```
select * from train_view
```

Results Explain Describe Saved SQL History

TRAIN_ID	TRAIN_NAME	DEPARTURE_STATION	ARRIVAL_STATION	DEPARTURE_TIME	ARRIVAL_TIME
1000	sakuntala express	yavatmal	wardha	10-SEP-22 02.30.00.000000 PM	10-SEP-22 04.30.00.000000 PM
1004	chennai express	mumbai	chennai	19-APR-20 06.30.00.000000 PM	22-APR-20 11.30.00.000000 AM
1009	rajdhani express	delhi	lucknow	11-AUG-11 08.20.00.000000 AM	22-AUG-11 01.30.00.000000 AM

3 rows returned in 0.00 seconds Download

14. View of 2 tables

Autocommit Rows 10 Save Run

```
create view demo as
select * from train, r
where train.train_id=1000;
```

Results Explain Describe Saved SQL History

```
View created.
```

0.00 seconds

Autocommit Rows 10 Save Run

```
select * from demo
```

Results Explain Describe Saved SQL History

TRAIN_ID	TRAIN_NAME	DEPARTURE_STATION	ARRIVAL_STATION	DEPARTURE_TIME	ARRIVAL_TIME	R_ID	P_ID	B_DATE	SEAT	STATUS
1000	sakuntala express	yavatmal	wardha	10-SEP-22 02.30.00.000000 PM	10-SEP-22 04.30.00.000000 PM	100	45	11/14/2003	26	booked
1000	sakuntala express	yavatmal	wardha	10-SEP-22 02.30.00.000000 PM	10-SEP-22 04.30.00.000000 PM	256	45	01/10/2013	34	waiting
1000	sakuntala express	yavatmal	wardha	10-SEP-22 02.30.00.000000 PM	10-SEP-22 04.30.00.000000 PM	114	45	12/16/2023	4	waiting

3 rows returned in 0.00 seconds Download

Assignment 2(b): DML

1. Insert into table train

Autocommit Rows 10 Save Run

```
INSERT INTO train (train_id, train_name, departure_station, arrival_station, departure_time, arrival_time)
VALUES (1016, 'raj express', 'mumbai', 'cochi', TO_TIMESTAMP('2023-09-11 08:20:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2023-09-24 01:30:00', 'YYYY-MM-DD HH24:MI:SS'));
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.01 seconds

2. Select from table

Autocommit Rows 10 Save Run

```
select * from train
```

Results Explain Describe Saved SQL History

TRAIN_ID	TRAIN_NAME	DEPARTURE_STATION	ARRIVAL_STATION	DEPARTURE_TIME	ARRIVAL_TIME
1000	sakuntala express	yavatmal	wardha	10-SEP-22 02.30.00.000000 PM	10-SEP-22 04.30.00.000000 PM
1004	chennai express	mumbai	chennai	19-APR-20 06.30.00.000000 PM	22-APR-20 11.30.00.000000 AM
1009	rajdhani express	delhi	lucknow	11-AUG-11 08.20.00.000000 AM	22-AUG-11 01.30.00.000000 AM
1016	raj express	mumbai	cochi	11-SEP-23 08.20.00.000000 AM	24-SEP-23 01.30.00.000000 AM

4 rows returned in 0.00 seconds Download

3. Select where like clause

Autocommit Rows 10 Save Run

```
select * from train where train_name like 'ra%'
```

Results Explain Describe Saved SQL History

TRAIN_ID	TRAIN_NAME	DEPARTURE_STATION	ARRIVAL_STATION	DEPARTURE_TIME	ARRIVAL_TIME
1009	rajdhani express	delhi	lucknow	11-AUG-11 08.20.00.000000 AM	22-AUG-11 01.30.00.000000 AM
1016	raj express	mumbai	cochi	11-SEP-23 08.20.00.000000 AM	24-SEP-23 01.30.00.000000 AM

2 rows returned in 0.00 seconds Download

4. Update

Autocommit Rows 10 Save Run

```
update train
set departure_station ='pune'
where train_id=1009
```

Results Explain Describe Saved SQL History

1 row(s) updated.

0.00 seconds

5. Delete

Autocommit Rows 10 Save Run

```
delete from train where train_id = 1000
```

Results Explain Describe Saved SQL History

1 row(s) deleted.

0.00 seconds

6. Group by clause

Autocommit Rows 10 Save Run

```
select count(train_id), train_name
from train
group by train_name
```

Results Explain Describe Saved SQL History

COUNT(TRAIN_ID)	TRAIN_NAME
1	rajdhani express
1	chennai express
1	raj express

3 rows returned in 0.00 seconds Download

7. Order by

```
select * from train
order by arrival_time
```

Results Explain Describe Saved SQL History

TRAIN_ID	TRAIN_NAME	DEPARTURE_STATION	ARRIVAL_STATION	DEPARTURE_TIME	ARRIVAL_TIME
1009	rajdhani express	pune	lucknow	11-AUG-11 08.20.00.000000 AM	22-AUG-11 01.30.00.000000 AM
1004	chennai express	mumbai	chennai	19-APR-20 06.30.00.000000 PM	22-APR-20 11.30.00.000000 AM
1016	raj express	mumbai	cochi	11-SEP-23 08.20.00.000000 AM	24-SEP-23 01.30.00.000000 AM

3 rows returned in 0.01 seconds Download

8. Count

Autocommit Rows 10 Save Run

```
select count(train_id)
from train
```

Results Explain Describe Saved SQL History

COUNT(TRAIN_ID)
3

1 rows returned in 0.00 seconds Download

9. Max

Autocommit Rows 10 Save Run

```
select max(train_id) as endid
from train
```

Results Explain Describe Saved SQL History

ENDID
1016

1 rows returned in 0.00 seconds Download

10. Min

Autocommit Rows 10 Save Run

```
select min(train_id) as endid
from train
```

Results Explain Describe Saved SQL History

ENDID
1004

1 rows returned in 0.00 seconds Download

Group A
Assignment Number 9
Database Connecvity

Problem Statement:

Write a program to implement MySQL/Oracle database connecvity with any front end language to implement Database navigaon operaons (add, delete, edit etc.)

PROGRAM

```
//Java program to illustrate Connecng to the Database

import java.sql.*;
public class connect { public stac void main(String args[])
{
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");

//Establishing Connecon

Connecon con= DriverManager.getConnecon("jdbc:oracle:thin:@localhost:1521 orcl","login!",
"pwd1");
if (con != null)
        System.out.println("Connected"); else
        System.out.println("Not Connected"); con.close();
}
catch(Excepon e)
{
System.out.println(e);
}
}
}
```

Implemenng Insert Statement

```
//Java program to illustrate inserng to the Database

import java.sql.*; public
class insert 1
{
public stac void main(String args[])
{ String id = "id";
 String pwd="pwd1";
 String fullname="geeks for geeks";
```

```
 String email="geeks@geeks.org";

try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connecon con- DriverManager.getConnecon("jdbc:oracle:thin:@localhost:1521 :orcl", "login1", "pwd1");
Statement stmt = con.createStatement();

//Inserng data in database

String q1 = "insert into userid values("+id+ "," +pwd+ "," +fullname+"", "+email+ "')"; int
x = stmt.executeUpdate(q1);
if (x>0)
            System.out.println("Successfully Inserted");
Else
             System.out.println("Insert Failed") con.close();
}
catch(Excepon e)
{
System.out.println(e);
}
}
}
```

Implemenng Update Statement

```
// Java program to illustrate updang the Database

import java.sql.*;
public class update1
{
public stac void main(String args[])
{
String id="id1";
String  pwd="pwd1"; String
newPwd="newpwd";
try
{
Class. For Name ("oracle.jdbc. driver. Oracle Driver")
Connecon  con=  DriverManager.getConnecon("jdbc:oracle: thin:@localhost:  1521:orcl",  "login1",
"pwd1");
Statement stmt = con.createStatement():

//Updang database

String q1="UPDATE userid set pwd="+newPwd+"WHERE id="+id+ " AND pwd="
+       pwd       +""       ;       int
x=stmt.executeUpdate(q1):
 if (x > 0)
```

```
            System.out.println("Password Successfully Updated");

Else
            Sysem.out.println("ERROR OCCURRED:(");  con.close();
}
catch(Excepon e)
{
System.out.println(e);
}
}
}
```

Implemenng Delete Statement

```
// Java program to illustrate deleng from Database

import java.sql.*;
public class delete { public stac void main(String args[])
{String id="id2"; String pwd="pwd2"; try
{
Class. for Name("oracle .jdbc .driver .Oracle Driver"); Connecon con= Driver Manager.get Connecon("jdbc:
oracle :thin:@localhost:1521:orcl", "login", "pwd1");
Statement stmt = con.createStatement();

// Deleng from database

String q1="DELETE from userid WHERE id="+id+"" AND pwd="" + pwd +""; int x = stmt .execute
Update(q1)
if (x > 0)
            System.out.println("One User Successfully Deleted"); else
            System.out.println("ERROR OCCURRED :("); con.close();
}
catch(Excepon e)
{
System.out.println(e);
}
}
}
```

Implemenng Select Statement

```
// Java program to illustrate selecng from Database

import java.sql.*;
public class select
{
public stac void main(String args[])
{
```

```
String id="id1"; String pwd="pwd1";
try
{
Class.forName("oracle.jdbc.driver OracleDriver");
Connecon con = DriverManager.getConnecon(" jdbc:oracle:thin:@localhost: 1521:orcl", "login1",
"pwd1");
Statement stmt= con.createStatement();
```

SELECT query

```
String q1="select * from userid WHERE id="+id+" AND pwd="+ pwd +" ResultSet rs stmt.executeQuery(q1);
if (rs.next())
{
System.out.println("User-Id:" + rs.getString(1));
System.out.println("Full Name:" + rs.getString(3)); System.out.println("E-mail
:"+rs.getString(4));
}
Else
{ System.out.println("No such user id is already registered");
}
con.close();
}
catch(Excepon e) {
System.out.println(e);
}
}
}
```

OUTPUT

User-id-id1
Full Name - geeks for geeks
E-mail - geeks@geeks.org