**ART GALLERY MANAGEMENT SYSTEM**

**UCS 310 Database Management System Project Report**

**END-Semester Evaluation**

**Submitted by:**

**(102117167) PRIYANKA BEDI**

**(102117168) SANEHA GARG**

**(102117166) AMISHA**

**(102117163) CHESHTA ARORA**

**BE Second Year, CSE**

**Submitted to:**

Dr. Geeta Kasana



**Computer Science and Engineering Department**

**TIET, Patiala**

**May 2023**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR.NO.** | **TOPIC** | **PG.NO.** |
| 1. | Introduction | 3 |
| 2. | ER Diagram | 5 |
| 3. | ER To Table | 6 |
| 4. | Normalization | 7 |
| 4.1. | First Normal Form(1NF) | 7 |
| 4.2. | Second Normal Form(2NF) | 8 |
| 4.3. | Third Normal Form(3NF) | 8 |
| 4.4. | Boyce-Coded Normal Form(BCNF) | 9 |
| 5. | SQL and PL/SQL Screenshots with Outputs | 9 |
| 5.1. | Creation and Insertion of Tables | 9 |
| 5.1.1. | Creation and Insertion of Gallery Table | 9 |
| 5.1.2. | Creation and Insertion of Exhibition Table | 10 |
| 5.1.3. | Creation and Insertion of Customer Table | 11 |
| 5.1.4. | Creation and Insertion of Artist Table | 12 |
| 5.1.5. | Creation and Insertion of Artwork Table | 13 |
| 5.1.6. | Creation and Insertion of Artist\_Styles Table | 14 |
| 5.1.7. | Creation and Insertion of Customer\_Contacts Table | 15 |
| 5.1.8. | Creation and Insertion of Buys Table | 16 |
| 5.2. | PL/SQL Blocks | 17 |
| 5.2.1. | Procedure to Display Artist with their Styles | 17 |
| 5.2.2. | Procedure to Display Customer with their Contacts | 18 |
| 5.2.3. | Procedure to Display a List of Artists with their Artworks | 20 |
| 5.2.4. | Procedure to Display Artist with their Artworks According to Specific Id | 22 |
| 5.2.5. | Cursor to Display a List of Gallery with their Artists | 23 |
| 5.2.6. | Cursor to Display Gallery with their Artists According to Specific Id | 24 |
| 5.2.7. | Cursor to Display a List of Exhibitions with their Gallery having Artists | 25 |
| 5.2.8. | Cursor to Display from which Gallery, Customer Bought which Artwork | 26 |
| 5.2.9. | Cursor to Display Relation Between Gallery, Exhibition, Artwork, Artist Tables | 28 |
| 5.2.10. | Cursor to Display Relation Between Gallery, Exhibition, Artwork, Artist, Customer Table | 29 |
| 5.2.11. | Procedure to Display Artist data According to Style Entered | 31 |
| 5.2.12. | Cursor to Display Artist and their Number of Artworks | 32 |
| 5.2.13. | Cursor to Display Customer and Number of Artworks they Bought | 33 |
| 5.2.14. | Cursor to Display a List of Customer and Artworks they Bought | 33 |
| 5.2.15. | Procedure to Delete Data from Table and Trigger | 35 |
| 5.2.16. | Procedure to Update Data and Trigger | 36 |
| 5.2.17. | Procedure to Insert Data and Trigger | 36 |
| 6. | Conclusion | 37 |

1. **INTRODUCTION**

An art gallery management database is an essential tool for art gallery owners and managers to organize and manage their artwork collection. The database can help you keep track of customers, artists, artworks, and exhibitions. It can also provide valuable insights into the performance of your gallery and help you make informed business decisions.

In this project, we have created a database that will serve as the backbone of your art gallery management system. The database will consist of tables that represent different entities, such as artwork, artists, clients, and exhibitions. SQL is used to create, update, and retrieve data from the database.

In this project, we have used PL/SQL to create stored procedures or functions that perform specific tasks, such as updating inventory levels or generating reports on sales and revenue. These stored procedures and functions can be called from within the database or from external applications, providing a convenient and efficient way to manage and manipulate data.

The goal of this project is to create a user-friendly and efficient art gallery management system that will improve the gallery's overall productivity and profitability. By the end of the project, you will have a functioning database that can be used to track and manage all aspects of your art gallery business.

Managing an art gallery can be a complex and challenging task. There are numerous aspects to consider, such organizing exhibitions and building relationships with artists and customers. An effective art gallery management database can be a powerful tool in streamlining these processes and enhancing the gallery's overall productivity and profitability.

The database will be designed to store and manage a wide range of information related to the gallery's operations. This information can include details on the artwork, such as the artist, title, type of art, year, and price, as well as information on the exhibitions, such as the start date and end date.

In addition, the database can help the gallery build relationships with its artists and customers. It can store details on artists, such as their name, birth place, and exhibition history. This can help the gallery to maintain a strong network of artists and ensure that they are represented in the best possible way.

The database can also be used to track customer information, such as their contact details, name, address, date of birth, and purchase history. This can help the gallery to build a better understanding of its customers and tailor its exhibitions and promotional activities to their interests.

Overall, an art gallery management database is an essential tool for art gallery owners and managers to help them manage and grow their business. By creating a well-designed and efficient database, you can streamline your gallery's operations, build stronger relationships with your artists and customers, and ultimately increase your profitability and success.

Commonly used statements are grouped into the following categories:

**Data Query Language (DQL)**

SELECT-Used to retrieve certain records from one or more tables.

**Data Manipulation Language (DML)**

INSERT - Used to create a record UPDATE - Used to change certain records. DELETE - Used to delete certain records.

**Data Definition Language (DDL)**

CREATE - Used to create a new table, a view of a table, or other object in database.

ALTER - Used to modify an existing database object, such as a table.

DROP - Used to delete an entire table, a view of a table or other object in the database.

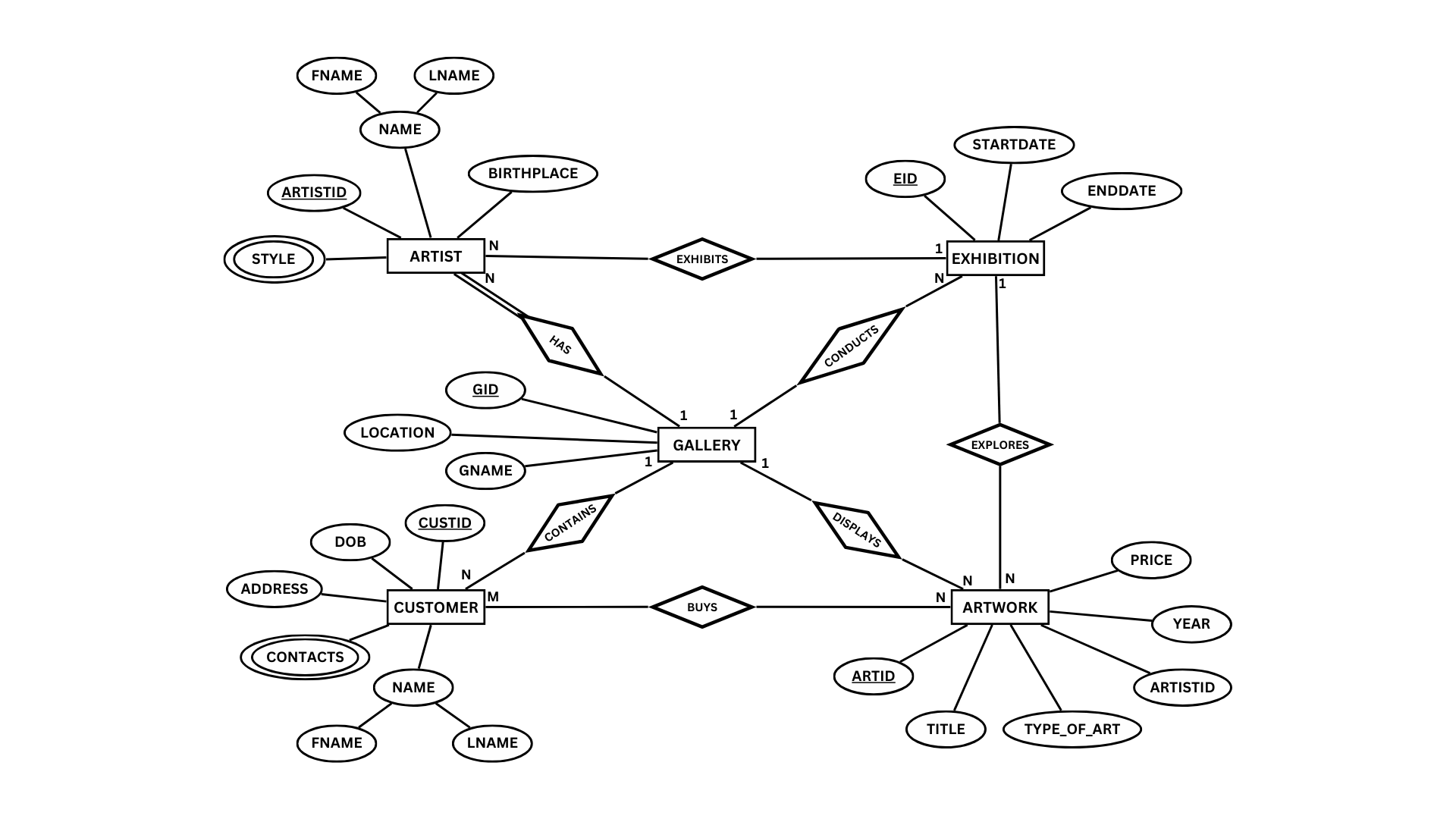
**SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Operating System | : 64bit WINDOWS Operating System, X64-based processor |
| Database | : MYSQL |

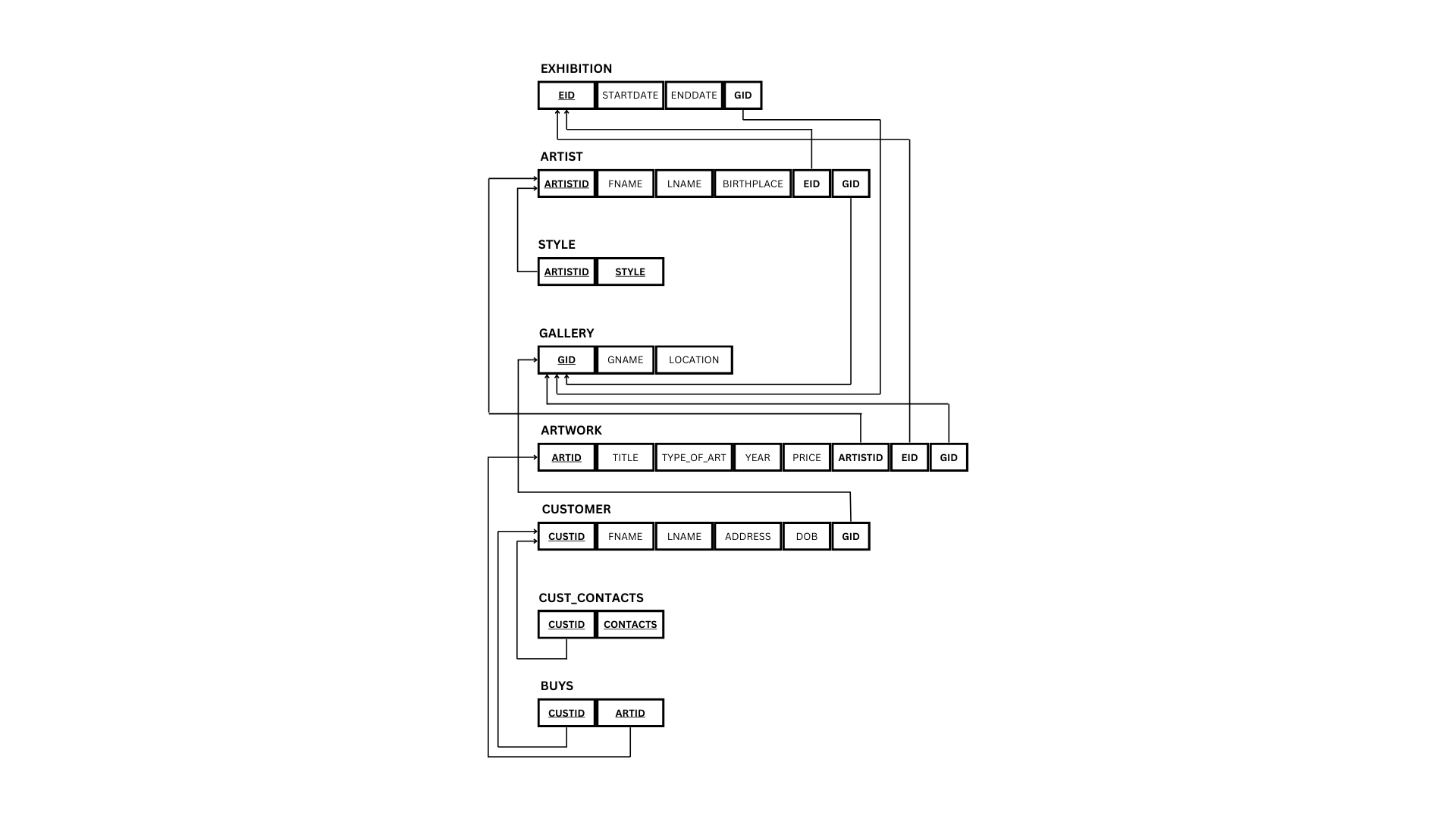
**HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Processor | : Intel Celeron CPU N3060 @1.60GHz or Above |
| RAM | : 4.00 GB or Above |
| Hard Disk | : 1 TB |
| Compact Disk | : CD-ROM, CD-R, CD-RW |
| Input device | : Keyboard |

1. **ER DIAGRAM**



1. **ER TO TABLE**



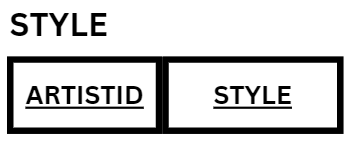
1. **NORMALIZATION**

**4.1. First Normal Form(1NF)**

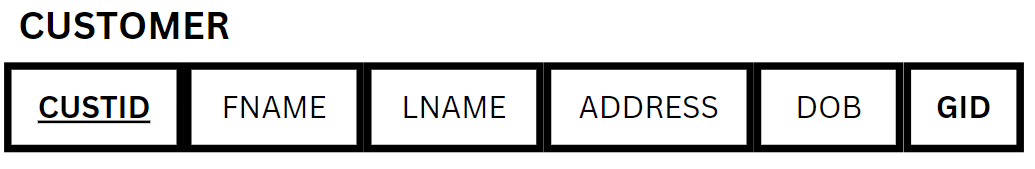
As per First normal form, no two rows of data must contain repeating group of information. Each set of columns must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that will distinguishes it as unique.

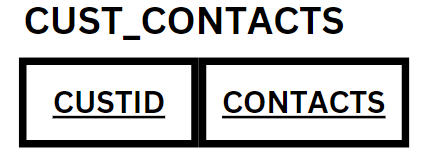
**Example:**





Here we have STYLE as multiple valued column so to keep it in 1NF we have created another table named STYLE which has a composite Primary key (ARTISTID, STYLE).





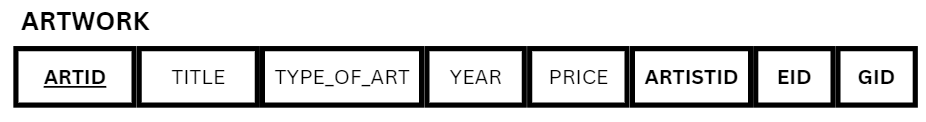
Here we have CONTACTS as multiple valued column so to keep it in 1NF we have created another table named CUST\_CONTACTS which has a composite Primary key (CUSTID, CONTACTS).

All the tables in the database are normalized to 1NF as all the attributes are atomic.

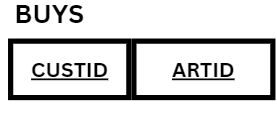
**4.2. Second Normal Form (2NF)**

A table is in 2NF if it is in 1NF and if all non-key attributes are fully functionally dependent on all of the key.

**Example:**



Here proper subsets of the candidate key (ARTID) do not identify non– prime attributes.



Here proper subsets of the candidate key (CUSTID, ARTID) do not identify non – prime attributes.

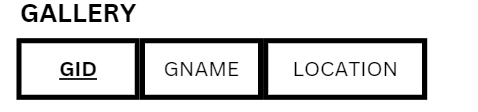
All the tables in the database are normalized to 2NF as they all are in 1NF and no proper subset of candidate key of the respective table identify respective non – prime attributes.

**4.3. Third Normal Form(3NF)**

A table is in 3NF if it is in 2NF and if it has no transitive dependency. X->Y, Y>Z, X>Z

According to CODD’s definition a relation schema R is in 3NF. It satisfies 2NF and no non-prime attribute of R is transitively dependent on the primary key.

**Example:**



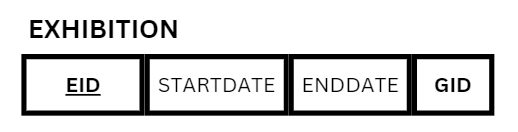
Here non - prime attribute (GNAME, LOCATION) do not identify any other non – prime attributes.

All the tables in the database are normalized to 3NF as they all are in 2NF and no non -prime of the respective table identify any other respective non – prime attributes.

**4.4. Boyce – Coded Normal Form(BCNF)**

The table must be in 3NF form and for any dependency X → Y, X must be a candidate key or super key. In other words, for dependency X → Y, if Y is a prime attribute, X cannot be a non-prime attribute.

**Example:**



All the tables in the database are normalized to BCNF as they all are in 3NF and for every functional dependency in the respective table has LHS either a candidate key or super key.

1. **SQL & PL/SQL SCREENSHOTS WITH OUTPUTS**
   1. **CREATION AND INSERTION OF TABLES**
      1. **CREATION AND INSERTION OF GALLERY TABLE**

CREATE TABLE GALLERY

(GID VARCHAR(20) PRIMARY KEY,

GNAME VARCHAR(20) NOT NULL,

LOCATION VARCHAR(20)NOT NULL);

INSERT INTO GALLERY VALUES('NG123','National Gallery', 'Washington');

INSERT INTO GALLERY VALUES('BM123','British Museum', 'London');

INSERT INTO GALLERY VALUES('JG123','Jahangir Gallery', 'Mumbai');

INSERT INTO GALLERY VALUES('TLM123','The Louvre Museum', 'Paris');

INSERT INTO GALLERY VALUES('MM123','Metropolitan Museum', 'New York');

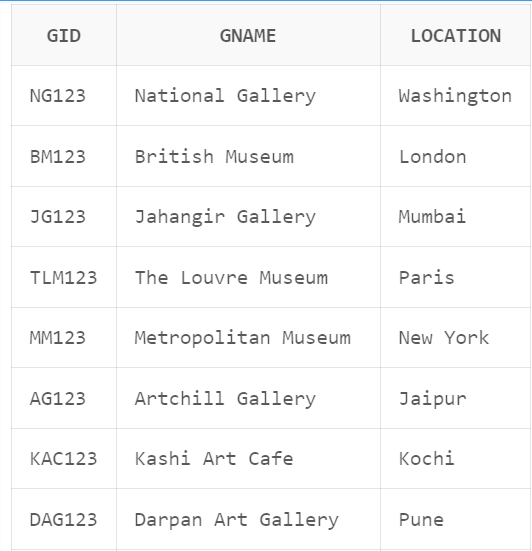
INSERT INTO GALLERY VALUES('AG123','Artchill Gallery', 'Jaipur');

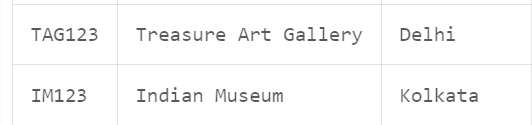
INSERT INTO GALLERY VALUES('KAC123','Kashi Art Cafe', 'Kochi');

INSERT INTO GALLERY VALUES('DAG123','Darpan Art Gallery', 'Pune');

INSERT INTO GALLERY VALUES('TAG123','Treasure Art Gallery', 'Delhi');

INSERT INTO GALLERY VALUES('IM123','Indian Museum', 'Kolkata');





* + 1. **CREATION AND INSERTION OF EXHIBITION TABLE**

CREATE TABLE EXHIBITION

(EID VARCHAR(20) PRIMARY KEY,

GID VARCHAR(20) DEFAULT NULL,

STARTDATE DATE DEFAULT NULL,

ENDDATE DATE DEFAULT NULL,

FOREIGN KEY(GID) REFERENCES GALLERY(GID) ON DELETE CASCADE, CHECK(ENDDATE>STARTDATE));

INSERT INTO EXHIBITION VALUES('G123','NG123','15-JUL-2022','30-JUL-2022');

INSERT INTO EXHIBITION VALUES('H123','BM123','21-AUG-2022','9-SEP-2022');

INSERT INTO EXHIBITION VALUES('I123','MM123','12-SEP-2022','25-SEP-2022');

INSERT INTO EXHIBITION VALUES('J123','TLM123','9-OCT-2022','20-OCT-2022');

INSERT INTO EXHIBITION VALUES('K123','JG123','7-NOV-2022','15-NOV-2022');

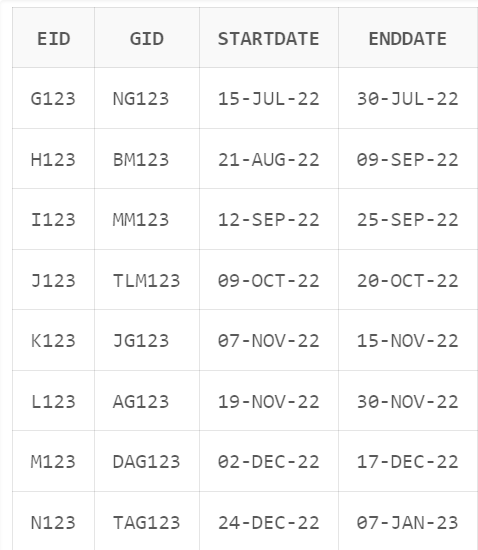
INSERT INTO EXHIBITION VALUES('L123','AG123','19-NOV-2022','30-NOV-2022');

INSERT INTO EXHIBITION VALUES('M123','DAG123','2-DEC-2022','17-DEC-2022');

INSERT INTO EXHIBITION VALUES('N123','TAG123','24-DEC-2022','7-JAN-2023');

INSERT INTO EXHIBITION VALUES('O123','IM123','11-JAN-2023','23-JAN-2023');

INSERT INTO EXHIBITION VALUES('P123','KAC123','26-JAN-2023','8-FEB-2023');





* + 1. **CREATION AND INSERTION OF CUSTOMER TABLE**

CREATE TABLE CUSTOMER

(CUSTID VARCHAR(20) PRIMARY KEY,

GID VARCHAR(20) DEFAULT NULL,

FNAME VARCHAR(20) DEFAULT NULL,

LNAME VARCHAR(20)DEFAULT NULL,

DOB DATE DEFAULT NULL,

ADDRESS VARCHAR(50)DEFAULT NULL,

FOREIGN KEY(GID) REFERENCES GALLERY(GID)ON DELETE CASCADE);

INSERT INTO CUSTOMER VALUES('AT2000','MM123','Akshay','Thakur','16-APR-2000','New York');

INSERT INTO CUSTOMER VALUES('AR1998','TLM123','Ashutosh','Ranjan','4-FEB-1998','Paris');

INSERT INTO CUSTOMER VALUES('AD1998','BM123','Ayush','Dhar','28-SEP-1998','London');

INSERT INTO CUSTOMER VALUES('AM1994','JG123','Avanish','Mehta','5-OCT-1994','Mumbai');

INSERT INTO CUSTOMER VALUES('JA1997','AG123','Jayant','Arora','7-NOV-1997','Jaipur');

INSERT INTO CUSTOMER VALUES('AG1980','DAG123','Ayan','Gupta','23-DEC-1980','Pune');

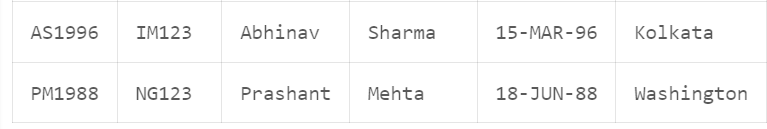
INSERT INTO CUSTOMER VALUES('AR1989','TAG123','Abhay','Raichand','17-JUL-1989','Delhi');

INSERT INTO CUSTOMER VALUES('RK1984','KAC123','Ranbeer','Kohli','8-APR-1984','Kochi');

INSERT INTO CUSTOMER VALUES('AS1996','IM123','Abhinav','Sharma','15-MAR-1996','Kolkata');

INSERT INTO CUSTOMER VALUES('PM1988','NG123','Prashant','Mehta','18-JUN-1988','Washington');





* + 1. **CREATION AND INSERTION OF ARTIST TABLE**

CREATE TABLE ARTIST

(ARTISTID VARCHAR(20) PRIMARY KEY,

GID VARCHAR(20)DEFAULT NULL,

CUSTID VARCHAR(20)DEFAULT NULL,

EID VARCHAR(20)DEFAULT NULL,

FNAME VARCHAR(20)NOT NULL,

LNAME VARCHAR(20)DEFAULT NULL,

BIRTHPLACE VARCHAR(20)DEFAULT NULL,

FOREIGN KEY(GID) REFERENCES GALLERY(GID)ON DELETE CASCADE,

FOREIGN KEY (CUSTID) REFERENCES CUSTOMER(CUSTID)ON DELETE CASCADE,

FOREIGN KEY(EID) REFERENCES EXHIBITION(EID)ON DELETE CASCADE);

INSERT INTO ARTIST VALUES('ART1','MM123','AT2000','O123','Georgia','O Keeffe','USA');

INSERT INTO ARTIST VALUES('ART2','TLM123','AR1998','P123','Pablo','Picasso','Spain');

INSERT INTO ARTIST VALUES('ART3','BM123','AG1980','N123','Rembrandt','van Rijn','Netherlands');

INSERT INTO ARTIST VALUES('ART4','IM123','AR1989','M123','Jamini','Roy','India');

INSERT INTO ARTIST VALUES('ART5','TAG123','RK1984','L123','John','Lewin','Austrailia');

INSERT INTO ARTIST VALUES('ART6','DAG123','AS1996','K123','Frans','Hals','Amsterdam');

INSERT INTO ARTIST VALUES('ART7','AG123','JA1997','J123','Martin','Aagaard','Norway');

INSERT INTO ARTIST VALUES('ART8','KAC123','AD1998','I123','Sieman','Allen','Africa');

INSERT INTO ARTIST VALUES('ART9','JG123','AM1994','H123','Theodore','Chasseriau','France');

INSERT INTO ARTIST VALUES('ART10','NG123','PM1988','G123','Leonardo','da Vinci','Italy');





* + 1. **CREATION AND INSERTION OF ARTWORK TABLE**

CREATE TABLE ARTWORK

(ARTID VARCHAR(20) PRIMARY KEY,

ARTISTID VARCHAR(20) DEFAULT NULL,

TITLE VARCHAR(20) NOT NULL,

YEAR INT DEFAULT NULL,

TYPE\_OF\_ART VARCHAR(20) DEFAULT NULL,

PRICE INT DEFAULT NULL,

EID VARCHAR(20)DEFAULT NULL,

GID VARCHAR(20)DEFAULT NULL,

FOREIGN KEY(EID) REFERENCES EXHIBITION(EID)ON DELETE CASCADE,

FOREIGN KEY(GID) REFERENCES GALLERY(GID)ON DELETE CASCADE,

FOREIGN KEY(ARTISTID) REFERENCES ARTIST(ARTISTID) ON DELETE CASCADE);

INSERT INTO ARTWORK VALUES('AW12','ART1','Mona Lisa','1503','Painting',10000000,'G123','NG123');

INSERT INTO ARTWORK VALUES('AW24','ART2','Poppies','1873','Painting',15000000,'H123','MM123');

INSERT INTO ARTWORK VALUES('AW36','ART3','Guernica','1937','Painting',25000000,'I123','TLM123');

INSERT INTO ARTWORK VALUES('AW48','ART4','The Last Supper','2003','Painting',9000000,'J123','BM123');

INSERT INTO ARTWORK VALUES('AW60','ART5','The Starry Night','1750','Sculpture',350000000,'K123','AG123');

INSERT INTO ARTWORK VALUES('AW66','ART6','Blessed Flair','1976','Painting',18000000,'L123','DAG123');

INSERT INTO ARTWORK VALUES('AW72','ART7','Frivolous Reality','1800','Sculpture',60000000,'M123','TAG123');

INSERT INTO ARTWORK VALUES('AW84','ART8','The Night Watch','2015','Painting',85000000,'N123','IM123');

INSERT INTO ARTWORK VALUES('AW90','ART9','Gracious Despair','1642','Painting',95000000,'O123','KAC123');

INSERT INTO ARTWORK VALUES('AW96','ART10','Two Sisters','2010','Sculpture',20000000,'P123','JG123');





* + 1. **CREATION AND INSERTION OF ARTIST\_STYLES TABLE**

CREATE TABLE ARTIST\_STYLES

(ARTISTID VARCHAR(20),

STYLE VARCHAR(20),

PRIMARY KEY(ARTISTID,STYLE),

FOREIGN KEY(ARTISTID) REFERENCES ARTIST(ARTISTID)ON DELETE CASCADE);

INSERT INTO ARTIST\_STYLES VALUES('ART1','Oil on Canvas');

INSERT INTO ARTIST\_STYLES VALUES('ART2','Analytic Cubism');

INSERT INTO ARTIST\_STYLES VALUES('ART2','Impressionist');

INSERT INTO ARTIST\_STYLES VALUES ('ART3','Oil Painting');

INSERT INTO ARTIST\_STYLES VALUES ('ART4','Symbolism');

INSERT INTO ARTIST\_STYLES VALUES ('ART5','Pop Art');

INSERT INTO ARTIST\_STYLES VALUES('ART5','High Renaissance');

INSERT INTO ARTIST\_STYLES VALUES ('ART6','Expressionism');

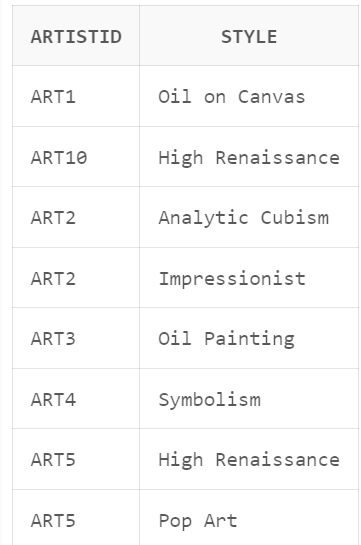
INSERT INTO ARTIST\_STYLES VALUES ('ART7','Realism');

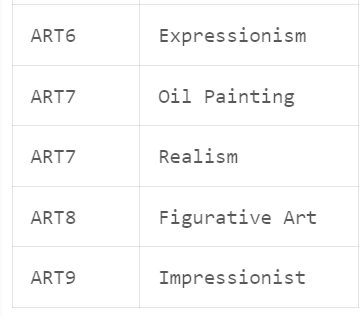
INSERT INTO ARTIST\_STYLES VALUES ('ART7','Oil Painting');

INSERT INTO ARTIST\_STYLES VALUES ('ART8','Figurative Art');

INSERT INTO ARTIST\_STYLES VALUES('ART9','Impressionist');

INSERT INTO ARTIST\_STYLES VALUES('ART10','High Renaissance');





* + 1. **CREATION AND INSERTION OF CUSTOMER\_CONTACTS TABLE**

CREATE TABLE CUST\_CONTACTS

(CUSTID VARCHAR(20),

CONTACTS NUMBER(10),

PRIMARY KEY(CUSTID,CONTACTS),

FOREIGN KEY(CUSTID)REFERENCES CUSTOMER(CUSTID)ON DELETE CASCADE);

INSERT INTO CUST\_CONTACTS VALUES('AT2000',9888776655);

INSERT INTO CUST\_CONTACTS VALUES('AR1998',9888779833);

INSERT INTO CUST\_CONTACTS VALUES('AD1998',9464105896);

INSERT INTO CUST\_CONTACTS VALUES('AD1998',9418687197);

INSERT INTO CUST\_CONTACTS VALUES('AM1994',8558903748);

INSERT INTO CUST\_CONTACTS VALUES('JA1997',8699592111);

INSERT INTO CUST\_CONTACTS VALUES('AG1980',9877041852);

INSERT INTO CUST\_CONTACTS VALUES('AG1980',7973301703);

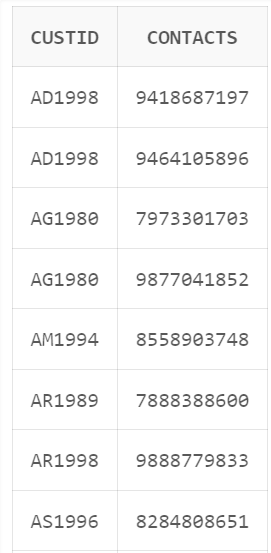
INSERT INTO CUST\_CONTACTS VALUES('AR1989',7888388600);

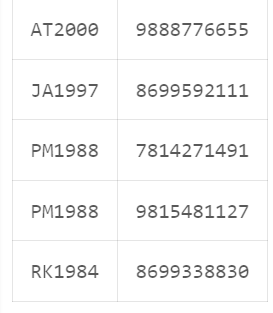
INSERT INTO CUST\_CONTACTS VALUES('RK1984',8699338830);

INSERT INTO CUST\_CONTACTS VALUES('AS1996',8284808651);

INSERT INTO CUST\_CONTACTS VALUES('PM1988',7814271491);

INSERT INTO CUST\_CONTACTS VALUES('PM1988',9815481127);





* + 1. **CREATION AND INSERTION OF BUYS TABLE**

CREATE TABLE BUYS

(CUSTID VARCHAR(20),

ARTID VARCHAR(20),

PRIMARY KEY(CUSTID,ARTID),

FOREIGN KEY(ARTID) REFERENCES ARTWORK(ARTID)ON DELETE CASCADE,

FOREIGN KEY(CUSTID)REFERENCES CUSTOMER(CUSTID) ON DELETE CASCADE);

INSERT INTO BUYS VALUES('AT2000','AW12');

INSERT INTO BUYS VALUES('AT2000','AW60');

INSERT INTO BUYS VALUES('AR1998','AW24');

INSERT INTO BUYS VALUES('AD1998','AW36');

INSERT INTO BUYS VALUES('AM1994','AW48');

INSERT INTO BUYS VALUES('JA1997','AW60');

INSERT INTO BUYS VALUES('AG1980','AW66');

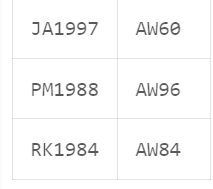
INSERT INTO BUYS VALUES('AR1989','AW72');

INSERT INTO BUYS VALUES('RK1984','AW84');

INSERT INTO BUYS VALUES('AS1996','AW90');

INSERT INTO BUYS VALUES('PM1988','AW96');





* 1. **PL/SQL BLOCKS**
     1. **PROCEDURE TO DISPLAY ARTIST WITH THEIR STYLES**

CREATE OR REPLACE PROCEDURE display\_artist\_styles IS

BEGIN

FOR artist\_rec IN (SELECT a.ARTISTID, a.FNAME, a.LNAME, a.BIRTHPLACE, s.STYLE

FROM ARTIST a

JOIN ARTIST\_STYLES s ON a.ARTISTID = s.ARTISTID

ORDER BY a.ARTISTID)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Artist ID: ' || artist\_rec.ARTISTID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || artist\_rec.FNAME || ' ' || artist\_rec.LNAME);

DBMS\_OUTPUT.PUT\_LINE('Birth Place: ' || artist\_rec.BIRTHPLACE);

DBMS\_OUTPUT.PUT\_LINE('Style: ' || artist\_rec.STYLE );

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('!! no such record exits !! ');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('!! TOO MANY ROWS RECORD !! ');

WHEN others THEN

DBMS\_OUTPUT.PUT\_LINE(' !! ERROR !! ');

END;

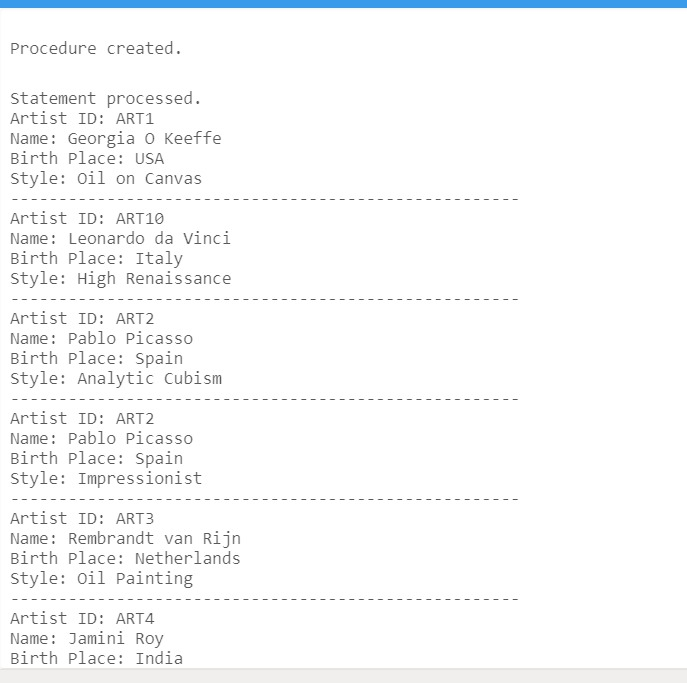
/

BEGIN

display\_artist\_styles;

END;

/





* + 1. **PROCEDURE TO DISPLAY CUSTOMER WITH THEIR CONTACTS**

CREATE OR REPLACE PROCEDURE display\_customer\_contacts IS

BEGIN

FOR cust\_rec IN (SELECT c.CUSTID, c.FNAME, c.LNAME, c.DOB, c.ADDRESS, ct.CONTACTS

FROM CUSTOMER c

LEFT JOIN CUST\_CONTACTS ct ON c.CUSTID = ct.CUSTID

ORDER BY c.CUSTID)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || cust\_rec.CUSTID);

DBMS\_OUTPUT.PUT\_LINE('Name: ' || cust\_rec.FNAME || ' ' || cust\_rec.LNAME);

DBMS\_OUTPUT.PUT\_LINE('Date of Birth: ' || cust\_rec.DOB);

DBMS\_OUTPUT.PUT\_LINE('Address: ' || cust\_rec.ADDRESS);

DBMS\_OUTPUT.PUT\_LINE('Contact: ' || cust\_rec.CONTACTS);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('!! no such record exits !! ');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('!! TOO MANY ROWS RECORD !! ');

WHEN others THEN

DBMS\_OUTPUT.PUT\_LINE(' !! ERROR !! ');

END;

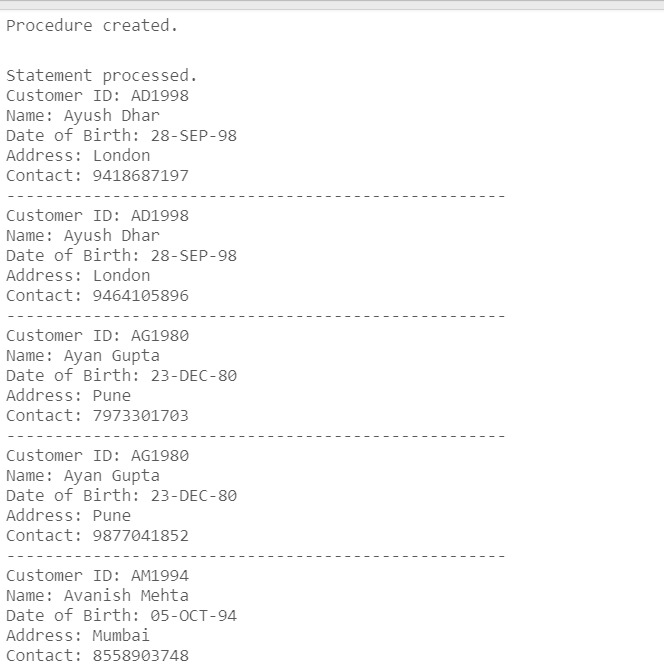
/

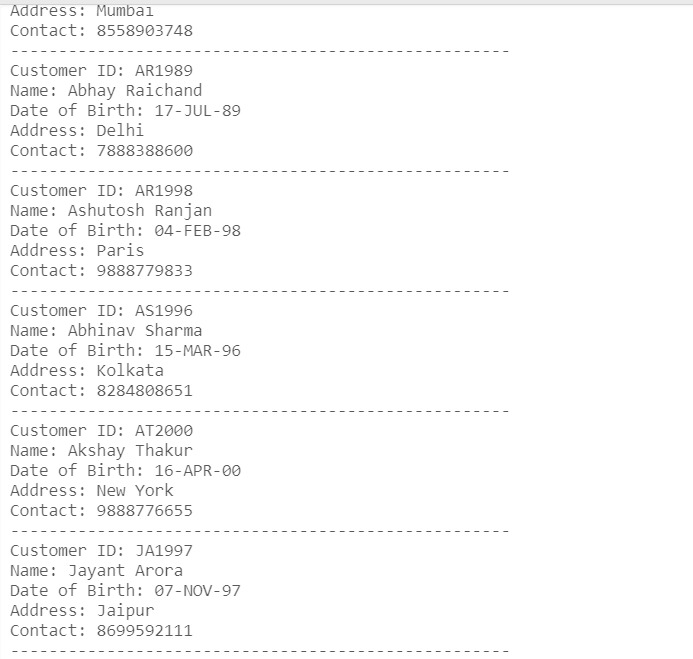
BEGIN

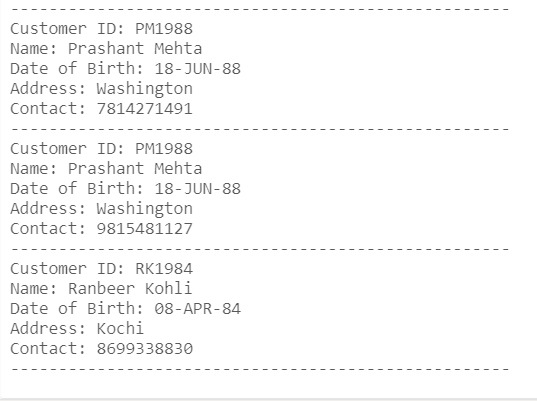
display\_customer\_contacts;

END;

/







* + 1. **PROCEDURE TO DISPLAY A LIST OF ARTISTS WITH THEIR ARTWORKS**

CREATE OR REPLACE PROCEDURE display\_all\_artist\_artworks IS

BEGIN

FOR artwork\_rec IN (SELECT A.ARTISTID , A.FNAME, A.LNAME, AR.ARTID, AR.TITLE, AR.TYPE\_OF\_ART, AR.YEAR, AR.PRICE FROM ARTIST A LEFT OUTER JOIN ARTWORK AR ON A.ARTISTID=AR.ARTISTID )

LOOP

DBMS\_OUTPUT.PUT\_LINE('Artist ID: ' || ARTWORK\_REC.ARTISTID);

DBMS\_OUTPUT.PUT\_LINE('Artist NAME: ' || ARTWORK\_REC.FNAME||' '||ARTWORK\_REC.LNAME);

DBMS\_OUTPUT.PUT\_LINE('Artwork ID: ' || artwork\_rec.ARTID);

DBMS\_OUTPUT.PUT\_LINE('Artwork Title: ' || artwork\_rec.title);

DBMS\_OUTPUT.PUT\_LINE('TYPE OF ART: ' || artwork\_rec.TYPE\_OF\_ART);

DBMS\_OUTPUT.PUT\_LINE('YEAR: ' || artwork\_rec.YEAR);

DBMS\_OUTPUT.PUT\_LINE('PRICE: ' || artwork\_rec.PRICE);

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('!! no such record exits !! ');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('!! TOO MANY ROWS RECORD !! ');

WHEN others THEN

DBMS\_OUTPUT.PUT\_LINE(' !! ERROR !! ');

END;

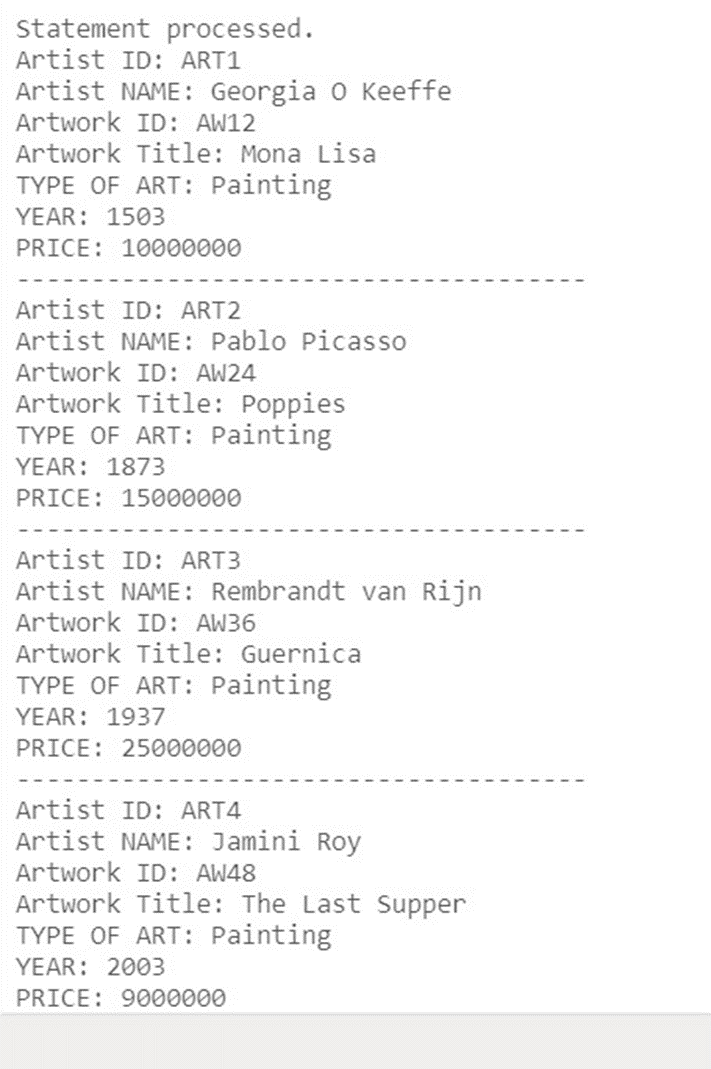
/

begin

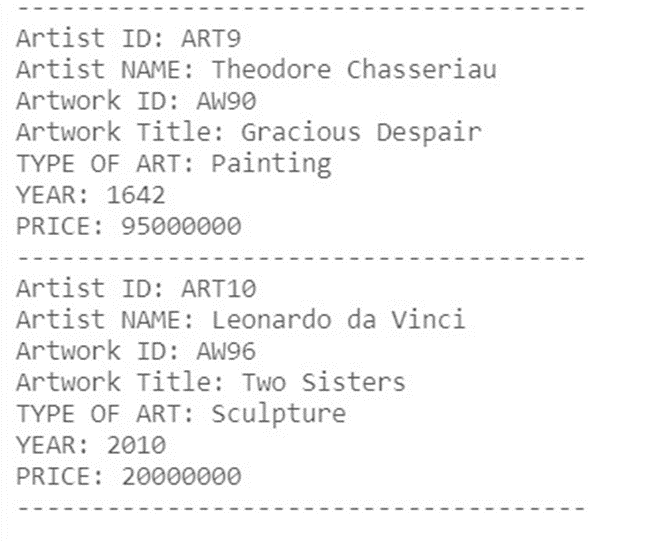
DISPLAY\_ALL\_ARTIST\_ARTWORKS;

end;

/







* + 1. **PROCEDURE TO DISPLAY ARTIST WITH THIER ARTWORKS ACCORDING TO SPECIFIC ID**

CREATE OR REPLACE PROCEDURE display\_artist\_artworks (artist\_id IN varchar) IS

BEGIN

FOR artwork\_rec IN (SELECT A.ARTISTID , A.FNAME, A.LNAME, AR.ARTID, AR.TITLE, AR.TYPE\_OF\_ART, AR.YEAR, AR.PRICE FROM ARTIST A LEFT OUTER JOIN ARTWORK AR ON A.ARTISTID=AR.ARTISTID WHERE A.ARTISTID=artist\_id )

LOOP

DBMS\_OUTPUT.PUT\_LINE('Artist ID: ' || ARTWORK\_REC.ARTISTID);

DBMS\_OUTPUT.PUT\_LINE('Artist NAME: ' || ARTWORK\_REC.FNAME||' '||ARTWORK\_REC.LNAME);

DBMS\_OUTPUT.PUT\_LINE('Artwork ID: ' || artwork\_rec.ARTID);

DBMS\_OUTPUT.PUT\_LINE('Artwork Title: ' || artwork\_rec.title);

DBMS\_OUTPUT.PUT\_LINE('TYPE OF ART: ' || artwork\_rec.TYPE\_OF\_ART);

DBMS\_OUTPUT.PUT\_LINE('YEAR: ' || artwork\_rec.YEAR);

DBMS\_OUTPUT.PUT\_LINE('PRICE: ' || artwork\_rec.PRICE);

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('!! no such record exits !! ');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('!! TOO MANY ROWS RECORD !! ');

WHEN others THEN

DBMS\_OUTPUT.PUT\_LINE(' !! ERROR !! ');

END;

/

declare

artist\_id varchar(20);

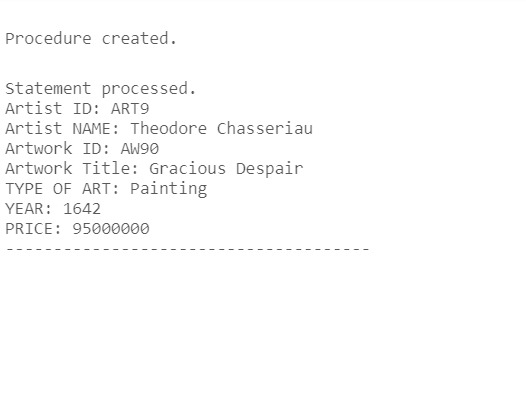
begin

artist\_id:='ART9';

display\_artist\_artworks(artist\_id);

end;

/



* + 1. **CURSOR TO DISPLAY A LIST OF GALLERY WITH THEIR ARTISTS**

DECLARE

CURSOR ARTIST\_INFO IS select G.GID,G.GNAME,G.LOCATION, A.ARTISTID,A.fname, A.LNAME from GALLERY G LEFT OUTER JOIN ARTIST A ON G.GID=A.GID;

REC1 ARTIST\_INFO%ROWTYPE;

BEGIN

OPEN ARTIST\_INFO;

LOOP

FETCH ARTIST\_INFO INTO REC1;

EXIT WHEN ARTIST\_INFO%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ARTISTS IN GALLERY: '||REC1.GID||' '||REC1.GNAME||' '||REC1.LOCATION);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('ARTIST ID: '||REC1.ARTISTID);

DBMS\_OUTPUT.PUT\_LINE('ARTIST NAME: '||REC1.fname||REC1.LNAME);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------');

END LOOP;

IF ARTIST\_INFO%NOTFOUND AND ARTIST\_INFO%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXCEPTION

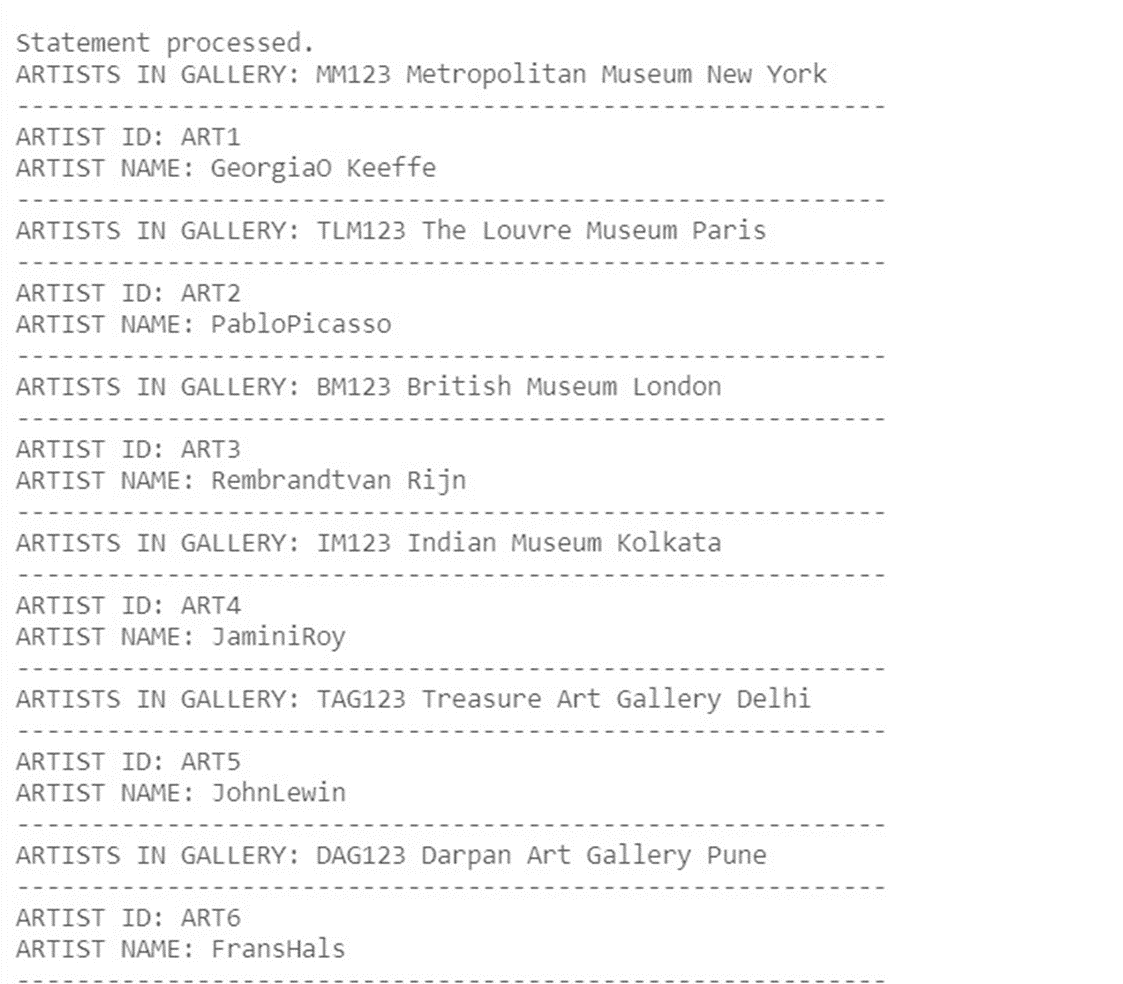
WHEN NO\_DATA\_FOUND THEN

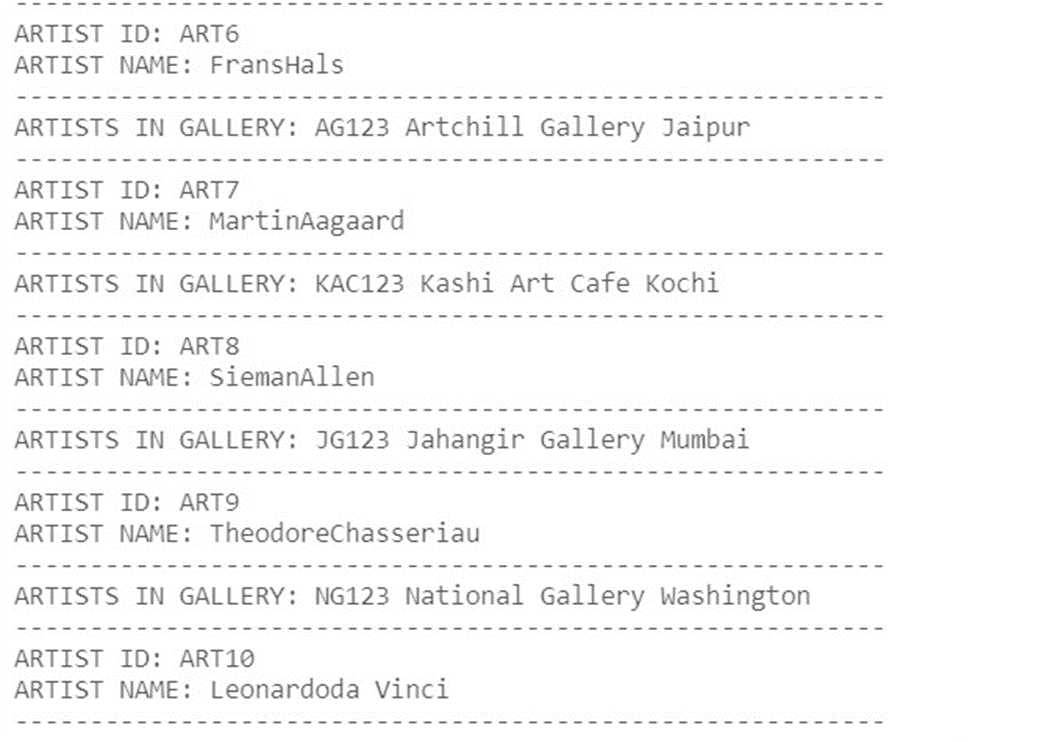
DBMS\_OUTPUT.PUT\_LINE('!!NO SUCH RECORD EXISTS!!');

CLOSE ARTIST\_INFO;

END;

/





* + 1. **CURSOR TO DISPLAY GALLERY WITH THEIR ARTISTS ACCORDING TO SPECIFIC ID**

DECLARE

ARTIST\_ID VARCHAR2(20);

CURSOR ARTIST\_INFO(ARTIST\_ID VARCHAR) IS select G.GID,G.GNAME,G.LOCATION, A.ARTISTID,A.fname, A.LNAME from GALLERY G LEFT OUTER JOIN ARTIST A ON G.GID=A.GID WHERE A.ARTISTID=ARTIST\_ID;

REC1 ARTIST\_INFO%ROWTYPE;

BEGIN

ARTIST\_ID:=&artist\_id;

OPEN ARTIST\_INFO(ARTIST\_ID);

LOOP

FETCH ARTIST\_INFO INTO REC1;

EXIT WHEN ARTIST\_INFO%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ARTISTS IN GALLERY: '||REC1.GID||' '||REC1.GNAME||' '||REC1.LOCATION);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('ARTIST ID: '||REC1.ARTISTID);

DBMS\_OUTPUT.PUT\_LINE('ARTIST NAME: '||REC1.fname||REC1.LNAME);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------------');

END LOOP;

IF ARTIST\_INFO%NOTFOUND AND ARTIST\_INFO%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXCEPTION

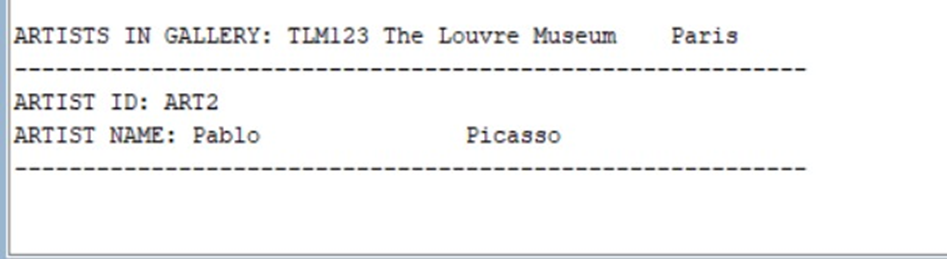
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('!!NO SUCH RECORD EXISTS!!');

CLOSE ARTIST\_INFO;

END;

/



* + 1. **CURSOR TO DISPLAY A LIST OF EXHIBITIONS WITH THEIR GALLERY HAVING ARTISTS**

DECLARE

CURSOR CUR\_EXHIBITION IS SELECT e.eid, e.startdate, e.enddate, a.artistid, a.fname,a.lname, g.gid, g.gname, g.location

FROM artist a

JOIN exhibition e ON e.eid = a.eid

JOIN GALLERY g ON g.gid = a.gid;

exhibition\_id EXHIBITION.eid%TYPE;

exhibition\_start\_date EXHIBITION.startdate%TYPE;

exhibition\_end\_date EXHIBITION.enddate%TYPE;

artist\_id ARTIST.artistid%TYPE;

artist\_fname ARTIST.fname%TYPE;

artist\_lname ARTIST.lname%TYPE;

gallery\_id GALLERY.gid%TYPE;

gallery\_name GALLERY.gname%TYPE;

gallery\_location GALLERY.location%TYPE;

BEGIN

OPEN CUR\_EXHIBITION;

LOOP

FETCH CUR\_EXHIBITION INTO

exhibition\_id, exhibition\_start\_date, exhibition\_end\_date, artist\_id,artist\_fname, artist\_lname, gallery\_id, gallery\_name, gallery\_location;

exit when cur\_exhibition%notfound;

DBMS\_OUTPUT.PUT\_LINE('Gallery ID: ' || gallery\_id || ', Gallery Name: ' || gallery\_name|| ', Gallery Location: ' || gallery\_location);

DBMS\_OUTPUT.PUT\_LINE('Exhibition ID: ' || exhibition\_id || ', Start Date: ' || exhibition\_start\_date || ', End Date: ' || exhibition\_end\_date);

DBMS\_OUTPUT.PUT\_LINE( 'Artist ID: ' || artist\_id || ', Artist Name: ' || artist\_fname || artist\_lname );

DBMS\_OUTPUT.PUT\_LINE('..............................................');

END LOOP;

IF CUR\_EXHIBITION%NOTFOUND AND CUR\_EXHIBITION%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXCEPTION

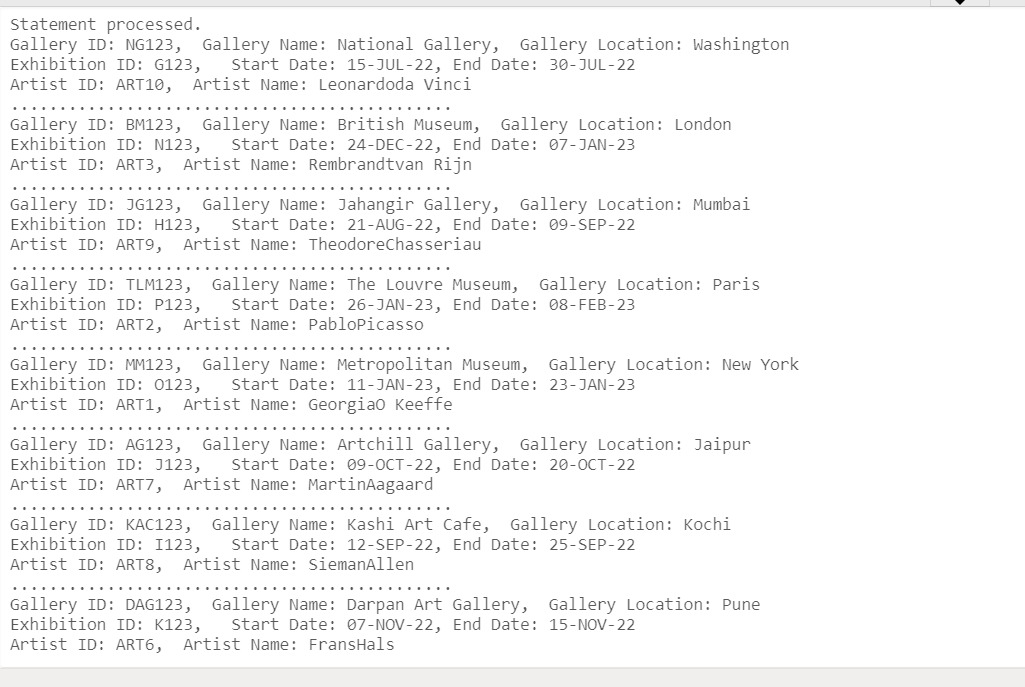
WHEN NO\_DATA\_FOUND THEN

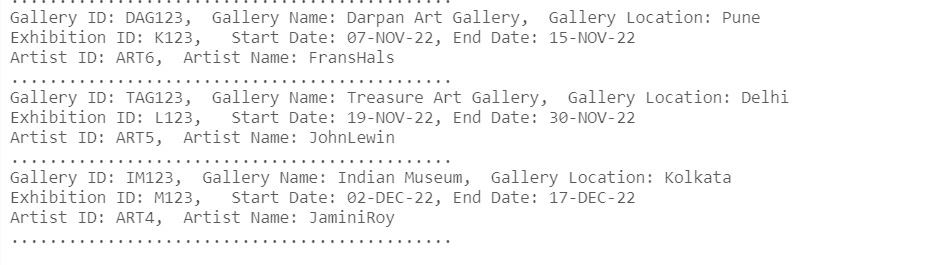
DBMS\_OUTPUT.PUT\_LINE('!!NO SUCH RECORD EXISTS!!');

CLOSE cur\_exhibition;

END;

/





* + 1. **CURSOR TO DISPLAY FROM WHICH GALLERY , CUSTOMER BOUGHT WHICH ARTWORK**

declare

CURSOR cur\_customer is

SELECT BUYS.CUSTID,customer.fname,customer.lname, ARTWORK.ARTID, GALLERY.GID, GALLERY.GNAME, GALLERY.LOCATION, ARTWORK.TITLE, ARTWORK.YEAR, ARTWORK.PRICE, ARTWORK.TYPE\_OF\_ART

FROM CUSTOMER

INNER JOIN BUYS ON BUYS.CUSTID=CUSTOMER.CUSTID

INNER JOIN ARTWORK ON BUYS.ARTID = ARTWORK.ARTID

INNER JOIN GALLERY ON ARTWORK.GID = GALLERY.GID;

custid VARCHAR(20);

cust\_fname varchar(20);

cust\_lname varchar(20);

artid VARCHAR(20);

gid VARCHAR(20);

gname VARCHAR(20);

title VARCHAR(20);

location VARCHAR(20);

year int;

price int;

type\_of\_art VARCHAR(20);

BEGIN

open cur\_customer;

LOOP

fetch cur\_customer into custid,cust\_fname,cust\_lname,artid,gid,gname,title,location,year,price,type\_of\_art;

EXIT WHEN CUR\_CUSTOMER%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID:' || custid||', CUSTOMER NAME: '||cust\_fname||' '||cust\_lname );

DBMS\_OUTPUT.PUT\_LINE('ART ID: ' || artid || ', Artwork Title: ' || title || ', Year: ' || year || ', Price: ' || price || ', Type of Art: ' || type\_of\_art);

DBMS\_OUTPUT.PUT\_LINE('GALLERY ID:' || gid || ', Gallery Name: ' || gname || ', Gallery Location: ' || location);

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------------------------------');

end loop;

IF cur\_customer%NOTFOUND AND cur\_customer%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXCEPTION

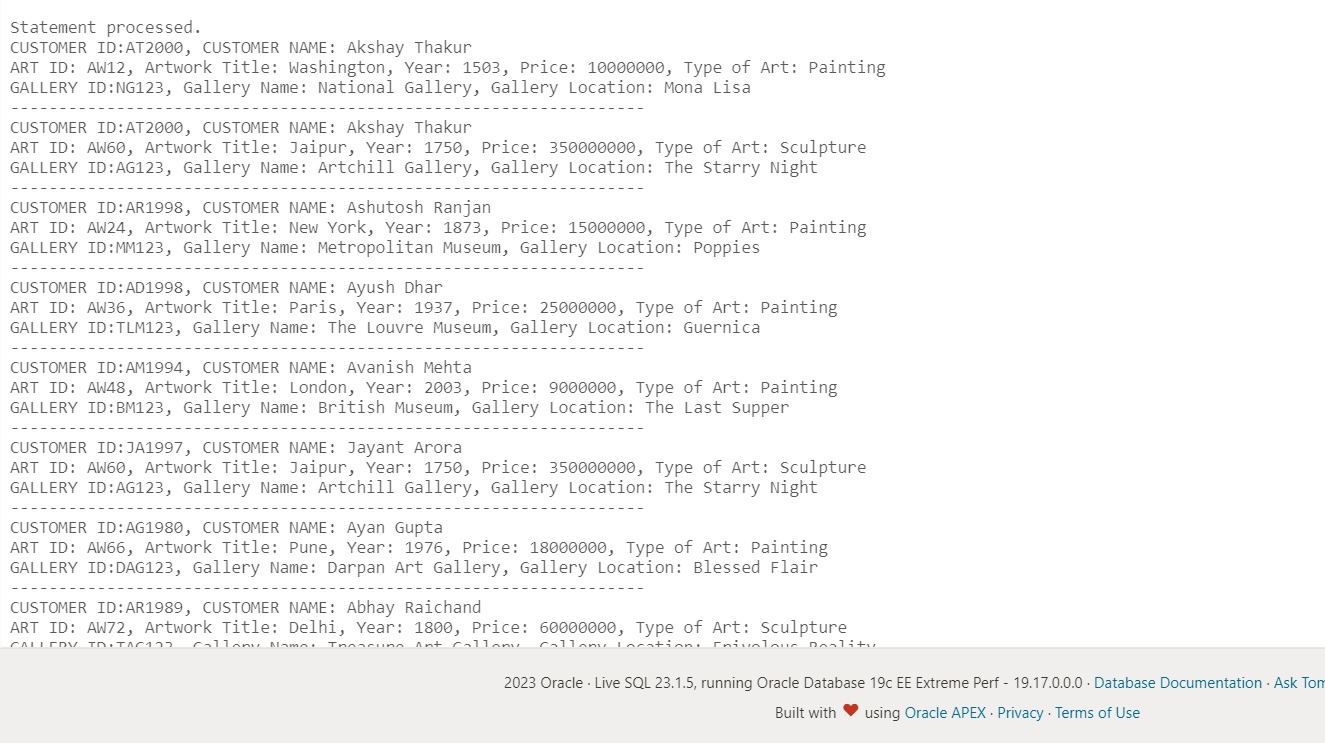
WHEN NO\_DATA\_FOUND THEN

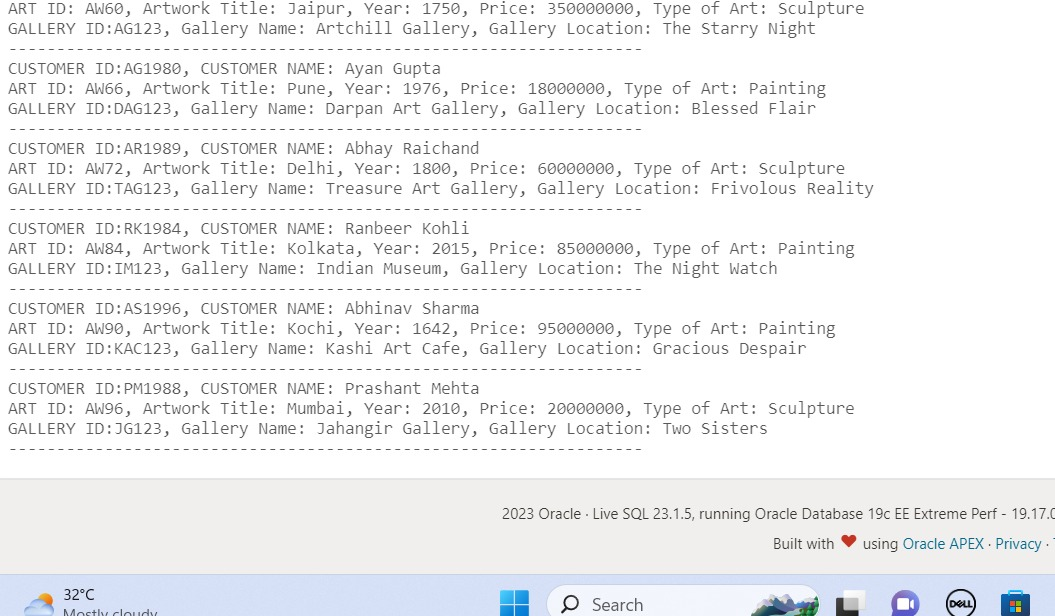
DBMS\_OUTPUT.PUT\_LINE('!!NO SUCH RECORD EXISTS!!');

CLOSE CUR\_CUSTOMER;

end;

/





* + 1. **CURSOR TO DISPLAY RELATION BETWEEN GALLERY , EXHIBITION , ARTWORK, ARTIST TABLES**

DECLARE

CURSOR myCursor is

SELECT G.GID, G.GNAME, G.LOCATION, E.EID, E.STARTDATE, E.ENDDATE,

A.ARTID, A.TITLE, A.YEAR, A.TYPE\_OF\_ART, A.PRICE,

AR. ARTISTID, AR.FNAME, AR.LNAME, AR.BIRTHPLACE

FROM GALLERY G

JOIN EXHIBITION E ON G.GID = E.GID

JOIN ARTWORK A ON E.EID = A.EID

JOIN ARTIST AR ON A.ARTISTID = AR.ARTISTID;

gid VARCHAR(20);

gname VARCHAR(20);

location VARCHAR(20);

eid VARCHAR(20);

startdate DATE;

enddate DATE;

artid VARCHAR(20);

title VARCHAR(20);

year INT;

type\_of\_art VARCHAR(20);

price INT;

artistid VARCHAR(20);

fname VARCHAR(20);

lname VARCHAR(20);

birthplace VARCHAR(20);

begin

OPEN myCursor;

loop

FETCH myCursor INTO gid, gname, location, eid, startdate, enddate, artid, title, year, type\_of\_art, price, artistid, fname, lname, birthplace;

EXIT WHEN myCursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE( 'Gallery ID: ' ||gid ||', Gallery Name: ' ||gname || ', Location: ' ||location);

DBMS\_OUTPUT.PUT\_LINE('Exhibition ID: ' ||eid || ', Start Date: ' ||startdate|| ', End Date: '||enddate);

DBMS\_OUTPUT.PUT\_LINE('Artwork ID: ' ||artid ||', Title: '||title || ', Year: '||year||', Type of Art: '||type\_of\_art ||', Price: ' ||price);

DBMS\_OUTPUT.PUT\_LINE( 'Artist ID: ' ||artistid ||', Name: ' ||fname || ' ' || lname || ', Birthplace: ' || birthplace);

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------');

end loop;

IF myCursor%NOTFOUND AND myCursor%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXCEPTION

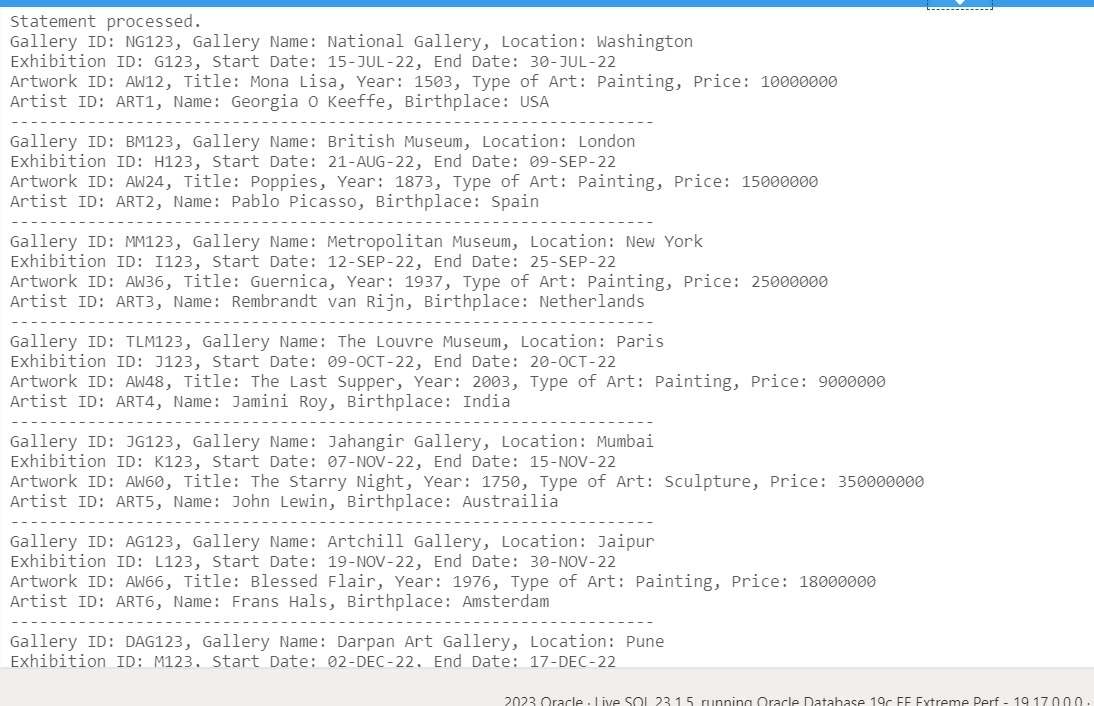
WHEN NO\_DATA\_FOUND THEN

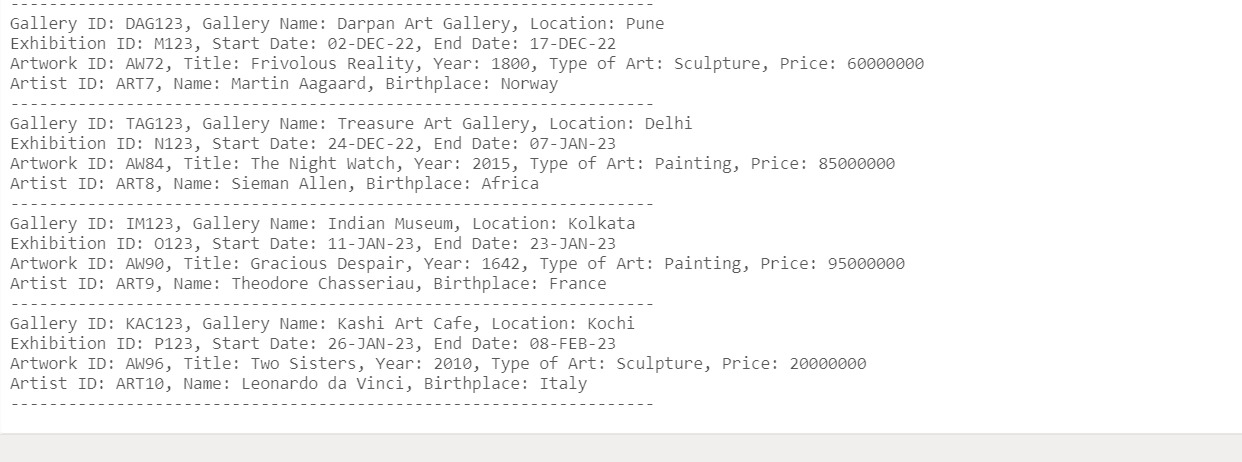
DBMS\_OUTPUT.PUT\_LINE('!!NO SUCH RECORD EXISTS!!');

CLOSE myCursor;

end;

/





* + 1. **CURSOR TO DISPLAY RELATION BETWEEN GALLERY , EXHIBITION , ARTWORK, ARTIST, CUSTOMER TABLES**

DECLARE

CURSOR myCursor1 is

SELECT G.GID, G.GNAME, G.LOCATION, E.EID, E.STARTDATE, E.ENDDATE,

A.ARTID, A.TITLE, A.YEAR, A.TYPE\_OF\_ART, A.PRICE,

AR. ARTISTID, AR.FNAME, AR.LNAME, AR.BIRTHPLACE,

C.CUSTID, C.FNAME, C.LNAME, C.ADDRESS, C.DOB

FROM GALLERY G

JOIN EXHIBITION E ON G.GID = E.GID

JOIN ARTWORK A ON E.EID = A.EID

JOIN ARTIST AR ON A.ARTISTID = AR.ARTISTID

JOIN CUSTOMER C ON G.GID= C.GID;

gid VARCHAR(20);

gname VARCHAR(20);

location VARCHAR(20);

eid VARCHAR(20);

startdate DATE;

enddate DATE;

artid VARCHAR(20);

title VARCHAR(20);

year INT;

type\_of\_art VARCHAR(20);

price INT;

artistid VARCHAR(20);

fname VARCHAR(20);

lname VARCHAR(20);

birthplace VARCHAR(20);

custid VARCHAR(20);

cfname VARCHAR(20);

clname VARCHAR(20);

caddress VARCHAR(20);

cdob date;

begin

OPEN myCursor1;

loop

FETCH myCursor1 INTO gid, gname, location, eid, startdate, enddate, artid, title, year, type\_of\_art, price, artistid, fname, lname, birthplace,custid,cfname,clname,caddress,cdob;

EXIT WHEN myCursor1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE( 'Gallery ID: ' ||gid ||', Gallery Name: ' ||gname || ', Location: ' ||location);

DBMS\_OUTPUT.PUT\_LINE('Exhibition ID: ' ||eid || ', Start Date: ' ||startdate|| ', End Date: '||enddate);

DBMS\_OUTPUT.PUT\_LINE('Artwork ID: ' ||artid ||', Title: '||title || ', Year: '||year||', Type of Art: '||type\_of\_art ||', Price: ' ||price);

DBMS\_OUTPUT.PUT\_LINE( 'Artist ID: ' ||artistid ||', Name: ' ||fname || ' ' || lname || ', Birthplace: ' || birthplace);

DBMS\_OUTPUT.PUT\_LINE( 'Customer ID: ' ||custid ||', Name: ' ||cfname || ' ' || clname || ', Address: ' || caddress || ', Date of Birth: ' || cdob);

DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------------------------------------------------------');

end loop;

IF myCursor1%NOTFOUND AND myCursor1%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('!!NO SUCH RECORD EXISTS!!');

CLOSE myCursor1;

end;

/





* + 1. **PROCEDURE TO DISPLAY ARTIST DATA ACCORDING TO STYLE ENTERED**

CREATE OR REPLACE PROCEDURE display\_artwork\_by\_style (p\_style IN artist\_styles.style%TYPE) AS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('ARTWORK STYLE : '||P\_STYLE);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------');

FOR a IN (

SELECT artwork.artid, artwork.title, artwork.year, artist.fname, artist.lname

FROM artwork

JOIN artist ON artwork.artistid = artist.artistid

JOIN artist\_styles ON artist\_styles.artistid = artist.artistid

WHERE artist\_styles.style = p\_style

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Artwork ID: ' || a.artid);

DBMS\_OUTPUT.PUT\_LINE('Title: ' || a.title);

DBMS\_OUTPUT.PUT\_LINE('Year Created: ' || a.year);

DBMS\_OUTPUT.PUT\_LINE('Artist Name: ' || a.fname || ' ' || a.lname);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------');

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('!! no such record exits !! ');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('!! TOO MANY ROWS RECORD !! ');

WHEN others THEN

DBMS\_OUTPUT.PUT\_LINE(' !! ERROR !! ');

END;

/

declare

style varchar(20);

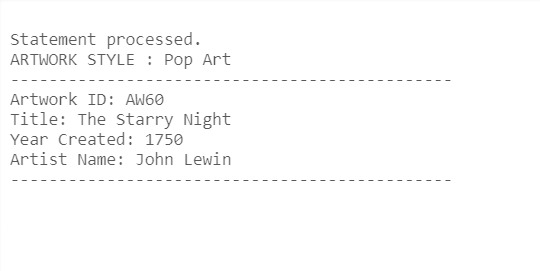
BEGIN

style:=&style;

display\_artwork\_by\_style(style);

END;

/



* + 1. **CURSOR TO DISPLAY ARTIST AND THEIR NUMBER OF ARTWORKS**

DECLARE

CURSOR c\_artist IS

SELECT A.FNAME,A.ARTISTID,A.LNAME, COUNT(B.ARTID) AS NUM\_ARTWORKS

FROM ARTIST A

LEFT OUTER JOIN ARTWORK B ON A.ARTISTID=B.ARTISTID GROUP BY A.ARTISTID,A.FNAME,A.LNAME;

REC2 C\_ARTIST%ROWTYPE;

BEGIN

OPEN c\_artist;

LOOP

fetch c\_artist into REC2;

exit when c\_artist%notfound;

DBMS\_OUTPUT.PUT\_LINE('Name: ' || REC2.fname ||' '|| REC2.lname || ', Artwork Count: ' || REC2.NUM\_ARTWORKS);

END LOOP;

close c\_artist;

END;

/



* + 1. **CURSOR TO DISPLAY CUSTOMER AND NUMBER OF ARTWORKS THEY BOUGHT**

DECLARE

CURSOR cust\_buys IS

SELECT C.FNAME,C.LNAME, COUNT(B.ARTID) AS NUM\_BUYS

FROM CUSTOMER C

LEFT JOIN BUYS B ON C.CUSTID = B.CUSTID

GROUP BY C.FNAME;

REC cust\_buys%ROWTYPE;

BEGIN

OPEN cust\_buys;

LOOP

FETCH cust\_buys INTO REC;

EXIT WHEN cust\_buys%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(REC.FNAME||' ' || REC.LNAME || ' - ' || REC.NUM\_BUYS || ' buys');

END LOOP;

IF cust\_buys%NOTFOUND AND cust\_buys%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

end if ;

EXCEPTION

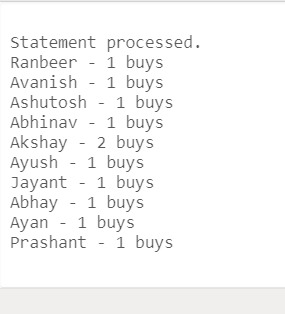
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('!! no such record exists !! ');

CLOSE cust\_buys;

END;

/



* + 1. **CURSOR TO DISPLAY A LIST OF CUSTOMER AND THE ARTWORKS THEY BOUGHT**

DECLARE

v\_custid VARCHAR(20);

v\_artid VARCHAR(20);

v\_title VARCHAR(20);

v\_fname VARCHAR(20);

v\_lname VARCHAR(20);

CURSOR c\_buy\_info IS

SELECT b.CUSTID, b.ARTID, a.TITLE, ar.FNAME, ar.LNAME

FROM BUYS b

JOIN ARTWORK a ON b.ARTID = a.ARTID

JOIN ARTIST ar ON a.ARTISTID = ar.ARTISTID

ORDER BY b.CUSTID;

BEGIN

OPEN c\_buy\_info ;

LOOP

FETCH c\_buy\_info INTO v\_custid , v\_artid , v\_title , v\_fname , v\_lname ;

EXIT WHEN c\_buy\_info%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE( 'CUSTOMER ID: ' ||v\_custid);

DBMS\_OUTPUT.PUT\_LINE('ARTWORK ID: ' ||v\_artid);

DBMS\_OUTPUT.PUT\_LINE('TITLE: ' || v\_title );

DBMS\_OUTPUT.PUT\_LINE( 'ARTIST NAME: ' ||v\_fname ||' ' || v\_lname);

DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------');

END LOOP;

IF c\_buy\_info%NOTFOUND AND c\_buy\_info%ROWCOUNT=0 THEN

RAISE NO\_DATA\_FOUND;

end if ;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

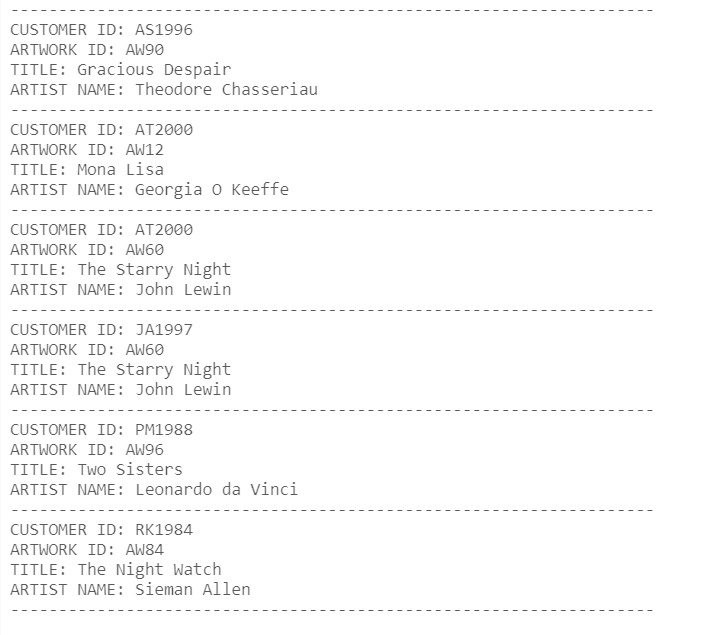
DBMS\_OUTPUT.PUT\_LINE('!! no such record exists !! ');

CLOSE c\_buy\_info ;

END;

/





* + 1. **PROCEDURE TO DELETE DATA FROM TABLE AND TRIGGER**

CREATE OR REPLACE TRIGGER DELETION

BEFORE DELETE ON GALLERY

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Data deleted');

DBMS\_OUTPUT.PUT\_LINE('OLD NAME '||:OLD.GNAME);

END;

/

create or replace procedure delete\_from\_GALLERY ( a in varchar) is

begin

delete from gallery where gid=a;

end;

/

DECLARE

GID VARCHAR(20);

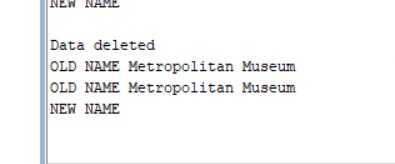
begin

GID:=&GID;

delete\_from\_GALLERY(GID);

END;

/



* + 1. **PROCEDURE TO UPDATE DATA AND TRIGGER**

CREATE OR REPLACE TRIGGER UPDATION

BEFORE UPDATE ON GALLERY

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Data Updated');

DBMS\_OUTPUT.PUT\_LINE('OLD NAME '||:OLD.GNAME);

DBMS\_OUTPUT.PUT\_LINE('NEW NAME '||:NEW.GNAME);

END;

/

create or replace procedure UPDATE\_GALLERY ( NAME in varchar, A IN VARCHAR) is

begin

UPDATE GALLERY SET GNAME=NAME WHERE GID=A;

end;

/

DECLARE

GNAME VARCHAR(20);

GID VARCHAR(20);

begin

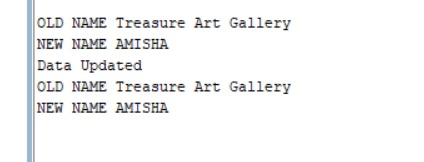
GID:=&GID;

GNAME:=&NEW\_GNAME;

UPDATE\_GALLERY(GNAME,GID);

END;

/



* + 1. **PROCEDURE TO INSERT DATA AND TRIGGER**

CREATE OR REPLACE TRIGGER INSERTION

BEFORE INSERT ON GALLERY

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Data Inserted');

DBMS\_OUTPUT.PUT\_LINE('ID '||:NEW.GID);

DBMS\_OUTPUT.PUT\_LINE('NAME '||:NEW.GNAME);

DBMS\_OUTPUT.PUT\_LINE('LOCATION '||:NEW.LOCATION);

END;

/

create or replace procedure INSERT\_IN\_GALLERY ( A in varchar,B IN VARCHAR,C IN VARCHAR) is

begin

INSERT INTO GALLERY VALUES(A,B,C);

end;

/

DECLARE

GID VARCHAR(20);

GNAME VARCHAR(20);

LOCATION VARCHAR(20);

BEGIN

GID:='AM1345';

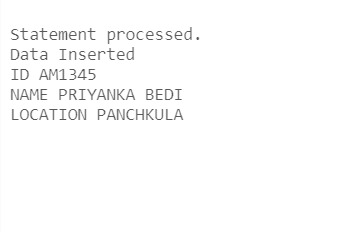
GNAME:='PRIYANKA BEDI';

LOCATION:='PANCHKULA';

INSERT\_IN\_GALLERY(GID,GNAME,LOCATION);

END;

/



1. **CONCLUSION**

An art gallery management database is a valuable tool for gallery owners and managers to manage the artwork, artists, customers, and exhibitions. A well-designed database can improve the organization and efficiency of the gallery, making it easier for gallery staff to find the information they need quickly and accurately.

One of the key benefits of an art gallery management database is the ability to manage artwork information. This can include information such as the artist's name, artwork title, type of art, year, and price.

In addition to managing artwork information, a database can also help manage artist information. This can include name, birth place, and exhibition history. With a database, gallery staff can easily keep track of which artists are represented by the gallery, which works belong to each artist, and which artists are currently exhibiting.

A database can also help manage customer information. This can include contact information, name, address, date of birth, and purchase history. With a database, gallery staff can easily track customer purchases, send targeted marketing communications, and provide personalized customer service.

In conclusion, an art gallery management database can greatly benefit a gallery by improving organization, efficiency, and decision-making. It provides a centralized location for important information, making it easier for gallery staff to access the information they need to do their jobs effectively. By implementing a database, gallery owners and managers can set their business up for success.