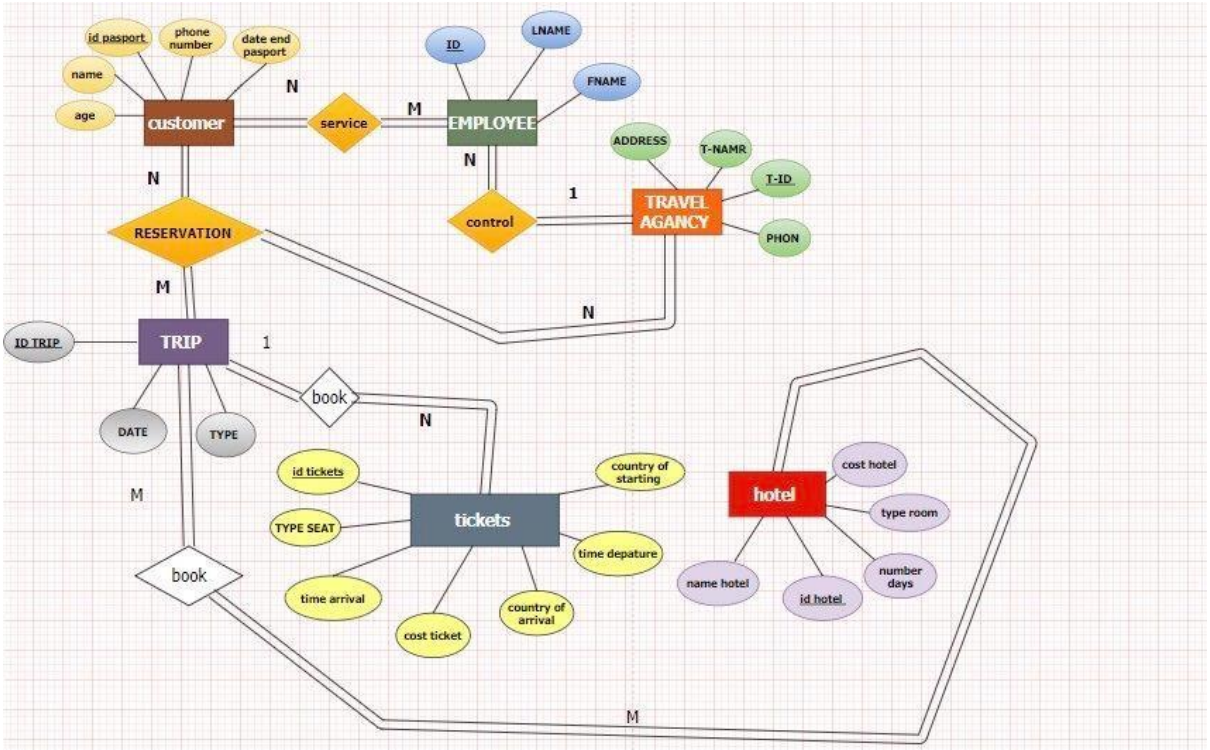
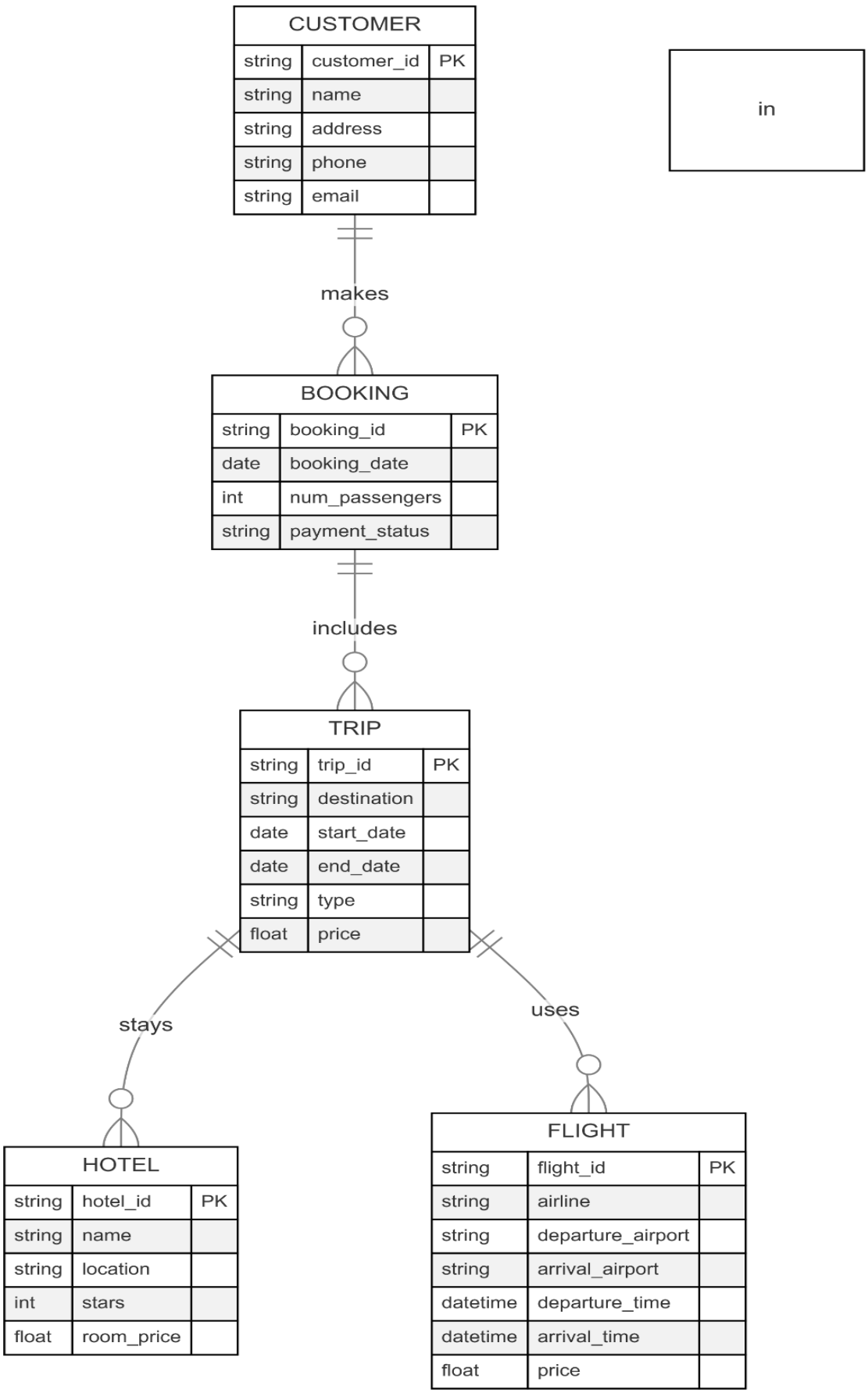
1.Draw an ER diagram and EER diagram and convert it into relational database and draw schema diagram.




CUSTOMER
string customer_id PK
string name
string address
string phone
string email
in
makes
BOOKING
string booking_id PK
date booking_date
int num_passengers
string payment_status
includes
TRIP
string trip_id PK
string destination
date start_date
date end_date
string type
float price
stays
uses
HOTEL
string hotel_id PK
string name
string location
int stars
float room_price
FLIGHT
string flight_id PK
string airline
string departure_airport
string arrival_airport
datetime departure_time
datetime arrival_time
float price

2. Write and execute basic SQL query- create, alter, insert, update and delete. (instructor should frame appropriate problem definition).

```
SQL> CREATE TABLE Tour_Packages (
  2      package_id NUMBER(5) PRIMARY KEY,
  3      package_name VARCHAR2(50) NOT NULL,
  4      destination VARCHAR2(50) NOT NULL,
  5      price NUMBER(10,2) NOT NULL,
  6      duration NUMBER(3) NOT NULL,
  7      availability VARCHAR2(10) CHECK (availability IN ('Available', 'Booked'))
  8  );

Table created.
```

```
SQL> CREATE TABLE Customers (
  2      customer_id NUMBER(5) PRIMARY KEY,
  3      name VARCHAR2(50) NOT NULL,
  4      contact VARCHAR2(15) UNIQUE NOT NULL,
  5      email VARCHAR2(50) UNIQUE NOT NULL,
  6      address VARCHAR2(100)
  7  );

Table created.
```

```
SQL> CREATE TABLE Bookings (
  2      booking_id NUMBER(5) PRIMARY KEY,
  3      customer_id NUMBER(5),
  4      package_id NUMBER(5),
  5      booking_date DATE DEFAULT SYSDATE,
  6      travel_date DATE NOT NULL,
  7      total_amount NUMBER(10,2) NOT NULL,
  8      status VARCHAR2(10) CHECK (status IN ('Confirmed', 'Pending', 'Canceled')),
  9      FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE CASCADE,
 10      FOREIGN KEY (package_id) REFERENCES Tour_Packages(package_id) ON DELETE CASCADE
 11  );

Table created.
```

```
SQL> CREATE TABLE Payments (
  2      payment_id NUMBER(5) PRIMARY KEY,
  3      booking_id NUMBER(5),
  4      payment_date DATE DEFAULT SYSDATE,
  5      amount NUMBER(10,2) NOT NULL,
  6      payment_mode VARCHAR2(20) CHECK (payment_mode IN ('Cash', 'Credit Card', 'Debit Card',
 'UPI', 'Net Banking')),
  7      FOREIGN KEY (booking_id) REFERENCES Bookings(booking_id) ON DELETE CASCADE
  8  );

Table created.
```

```
SQL> INSERT INTO Tour_Packages (package_id, package_name, destination, price, duration, availability) VALUES
  2  (101, 'Goa Adventure', 'Goa', 25000, 5, 'Available');

1 row created.

SQL> INSERT INTO Tour_Packages (package_id, package_name, destination, price, duration, availability) VALUES
  2  (102, 'Himalayan Escape', 'Manali', 30000, 6, 'Booked');

1 row created.

SQL> INSERT INTO Tour_Packages (package_id, package_name, destination, price, duration, availability) VALUES
  2  (103, 'Royal Rajasthan', 'Jaipur', 20000, 4, 'Available');

1 row created.

SQL> INSERT INTO Tour_Packages (package_id, package_name, destination, price, duration, availability) VALUES
  2  (104, 'Kerala Backwaters', 'Alleppey', 28000, 5, 'Available');

1 row created.

SQL> INSERT INTO Tour_Packages (package_id, package_name, destination, price, duration, availability) VALUES
  2  (105, 'Andaman Bliss', 'Port Blair', 35000, 7, 'Booked');

1 row created.
```

```
SQL> INSERT INTO Customers (customer_id, name, contact, email, address) VALUES
  2  (201, 'Rohan Sharma', '9876543210', 'rohan.sharma@email.com', 'Delhi');

1 row created.

SQL> INSERT INTO Customers (customer_id, name, contact, email, address) VALUES
  2  (202, 'Neha Verma', '8765432109', 'neha.verma@email.com', 'Mumbai');

1 row created.

SQL> INSERT INTO Customers (customer_id, name, contact, email, address) VALUES
  2  (203, 'Akash Mehta', '7654321098', 'akash.mehta@email.com', 'Bangalore');

1 row created.

SQL> INSERT INTO Customers (customer_id, name, contact, email, address) VALUES
  2  (204, 'Pooja Sinha', '6543210987', 'pooja.sinha@email.com', 'Kolkata');

1 row created.

SQL> INSERT INTO Customers (customer_id, name, contact, email, address) VALUES
  2  (205, 'Rajiv Kapoor', '5432109876', 'rajiv.kapoor@email.com', 'Chennai');
```

```
SQL> INSERT INTO Bookings (booking_id, customer_id, package_id, booking_date, travel_date, total_amount, status) VALUES
  2  (301, 201, 101, TO_DATE('2024-11-01', 'YYYY-MM-DD'), TO_DATE('2024-12-05', 'YYYY-MM-DD'), 25000, 'Confirmed');

1 row created.

SQL> INSERT INTO Bookings (booking_id, customer_id, package_id, booking_date, travel_date, total_amount, status) VALUES
  2  (302, 202, 103, TO_DATE('2024-10-15', 'YYYY-MM-DD'), TO_DATE('2024-11-20', 'YYYY-MM-DD'), 20000, 'Pending');

1 row created.

SQL> INSERT INTO Bookings (booking_id, customer_id, package_id, booking_date, travel_date, total_amount, status) VALUES
  2  (303, 203, 102, TO_DATE('2024-09-25', 'YYYY-MM-DD'), TO_DATE('2024-10-30', 'YYYY-MM-DD'), 30000, 'Canceled');

1 row created.

SQL> INSERT INTO Bookings (booking_id, customer_id, package_id, booking_date, travel_date, total_amount, status) VALUES
  2  (304, 204, 104, TO_DATE('2024-08-10', 'YYYY-MM-DD'), TO_DATE('2024-09-15', 'YYYY-MM-DD'), 28000, 'Confirmed');

1 row created.

SQL> INSERT INTO Bookings (booking_id, customer_id, package_id, booking_date, travel_date, total_amount, status) VALUES
  2  (305, 205, 105, TO_DATE('2024-07-05', 'YYYY-MM-DD'), TO_DATE('2024-08-10', 'YYYY-MM-DD'), 35000, 'Confirmed');

1 row created.
```

```
SQL> INSERT INTO Payments (payment_id, booking_id, payment_date, amount, payment_mode) VALUES
  2  (401, 301, TO_DATE('2024-11-02', 'YYYY-MM-DD'), 25000, 'Credit Card');

1 row created.

SQL> INSERT INTO Payments (payment_id, booking_id, payment_date, amount, payment_mode) VALUES
  2  (402, 302, TO_DATE('2024-10-16', 'YYYY-MM-DD'), 10000, 'UPI');

1 row created.

SQL> INSERT INTO Payments (payment_id, booking_id, payment_date, amount, payment_mode) VALUES
  2  (403, 303, TO_DATE('2024-09-26', 'YYYY-MM-DD'), 30000, 'Net Banking');

1 row created.

SQL> INSERT INTO Payments (payment_id, booking_id, payment_date, amount, payment_mode) VALUES
  2  (404, 304, TO_DATE('2024-08-11', 'YYYY-MM-DD'), 28000, 'Debit Card');

1 row created.

SQL> INSERT INTO Payments (payment_id, booking_id, payment_date, amount, payment_mode) VALUES
  2  (405, 305, TO_DATE('2024-07-06', 'YYYY-MM-DD'), 35000, 'Cash');

1 row created.
```

3.Write and execute SQL functions- aggregate, numeric, date, string, and conversion.

```
SQL> SELECT SUM(Price) AS Total_Revenue FROM Tickets;

TOTAL_REVENUE
-------------
           55

SQL> SELECT COUNT(Attendee_ID) AS Total_Attendees FROM Attendee;

TOTAL_ATTENDEES
---------------
              1

SQL> SELECT AVG(Price) AS Avg_Ticket_Price FROM Tickets;

AVG_TICKET_PRICE
----------------
              55
```

```
SQL> SELECT ROUND(Price) AS Rounded_Price FROM Tickets;

ROUNDED_PRICE
-------------
           55

SQL> SELECT FLOOR(Price) AS Floor_Price FROM Tickets;

FLOOR_PRICE
-----------
         55

SQL> SELECT Price, MOD(Price, 5) AS Remainder FROM Tickets;

     PRICE  REMAINDER
---------- ----------
        55          0
```

```
SQL> SELECT Event_ID, EXTRACT(YEAR FROM Event_Date) AS Event_Year FROM Event;

  EVENT_ID EVENT_YEAR
---------- ----------
         1       2025

SQL> SELECT Order_ID, Order_Time, Order_Time + INTERVAL '10' DAY AS New_Order_Time FROM Order_Details;

  ORDER_ID
----------
ORDER_TIME
---------------------------------------------------------------------------
NEW_ORDER_TIME
---------------------------------------------------------------------------
         1
10-JUN-25 02.00.00.000000 PM
20-JUN-25 02.00.00.000000000 PM


SQL> SELECT e.Event_ID, e.Event_Date, o.Order_Time,
  2         e.Event_Date - CAST(o.Order_Time AS DATE) AS Days_To_Event
  3  FROM Event e, Order_Details o;

  EVENT_ID EVENT_DAT
---------- ---------
ORDER_TIME
---------------------------------------------------------------------------
DAYS_TO_EVENT
-------------
         1 15-JUN-25
```

4. Write and execute SQL queries- Operators (and, or, not, like, between, in)

```
SQL> SELECT name, contact
  2  FROM Customers
  3  WHERE customer_id IN (SELECT customer_id FROM Bookings);

NAME                                               CONTACT
-------------------------------------------------- ----------
Rohan Sharma                                       9876543210
Neha Verma                                         8765432109
Akash Mehta                                        7654321098
Pooja Sinha                                        6543210987
Rajiv Kapoor                                       5432109876

SQL> SELECT name, contact
  2  FROM Customers
  3  WHERE customer_id NOT IN (SELECT customer_id FROM Bookings);

no rows selected

SQL> SELECT booking_id, total_amount
  2  FROM Bookings
  3  WHERE total_amount > ANY (SELECT amount FROM Payments);

BOOKING_ID TOTAL_AMOUNT
---------- ------------
       305        35000
       303        30000
       304        28000
       301        25000
       302        20000

SQL> SELECT booking_id, total_amount
  2  FROM Bookings
  3  WHERE total_amount > ALL (SELECT amount FROM Payments);

no rows selected
```

```
SQL> SELECT package_name, destination
  2  FROM Tour_Packages TP
  3  WHERE EXISTS (SELECT 1 FROM Bookings B WHERE B.package_id = TP.package_id);

PACKAGE_NAME
--------------------------------------------------
DESTINATION
--------------------------------------------------
Goa Adventure
Goa

Royal Rajasthan
Jaipur

Himalayan Escape
Manali


PACKAGE_NAME
--------------------------------------------------
DESTINATION
--------------------------------------------------
Kerala Backwaters
Alleppey

Andaman Bliss
Port Blair
```

5.Write and execute SQL queries- subqueries, joins.

```
SQL> SELECT C.customer_id, C.name, B.booking_id, B.total_amount
  2  FROM Customers C
  3  FULL OUTER JOIN Bookings B
  4  ON C.customer_id = B.customer_id;

CUSTOMER_ID NAME                                               BOOKING_ID
----------- -------------------------------------------------- ----------
TOTAL_AMOUNT
------------
        201 Rohan Sharma                                              301
       25000

        202 Neha Verma                                                302
       20000

        203 Akash Mehta                                               303
       30000


CUSTOMER_ID NAME                                               BOOKING_ID
----------- -------------------------------------------------- ----------
TOTAL_AMOUNT
------------
        204 Pooja Sinha                                               304
       28000

        205 Rajiv Kapoor                                              305
       35000
```

```
SQL> SELECT C.customer_id, C.name, B.booking_id, B.total_amount
  2  FROM Customers C
  3  LEFT JOIN Bookings B
  4  ON C.customer_id = B.customer_id;

CUSTOMER_ID NAME                                               BOOKING_ID
----------- -------------------------------------------------- ----------
TOTAL_AMOUNT
------------
        201 Rohan Sharma                                              301
       25000

        202 Neha Verma                                                302
       20000

        203 Akash Mehta                                               303
       30000


CUSTOMER_ID NAME                                               BOOKING_ID
----------- -------------------------------------------------- ----------
TOTAL_AMOUNT
------------
        204 Pooja Sinha                                               304
       28000

        205 Rajiv Kapoor                                              305
       35000


SQL> SELECT C.customer_id, C.name, B.booking_id, B.total_amount
  2  FROM Customers C
  3  NATURAL JOIN Bookings B;
SELECT C.customer_id, C.name, B.booking_id, B.total_amount
       *
ERROR at line 1:
ORA-25155: column used in NATURAL join cannot have qualifier
```

6. Write and execute basic Pl/SQL programs - simple program, condition statements and loops.

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2    grade CHAR(1);
  3  BEGIN
  4    grade := 'B';
  5
  6    IF grade = 'A' THEN
  7      DBMS_OUTPUT.PUT_LINE('Excellent');
  8    ELSIF grade = 'B' THEN
  9      DBMS_OUTPUT.PUT_LINE('Very Good');
 10    ELSIF grade = 'C' THEN
 11      DBMS_OUTPUT.PUT_LINE('Good');
 12    ELSIF grade = 'D' THEN
 13      DBMS_OUTPUT. PUT_LINE('Fair');
 14    ELSIF grade = 'F' THEN
 15      DBMS_OUTPUT.PUT_LINE('Poor');
 16    ELSE
 17      DBMS_OUTPUT.PUT_LINE('No such grade');
 18    END IF;
 19  END;
 20  /
Very Good

PL/SQL procedure successfully completed.
```

7.Write and execute Pl/SQL function to print /return binary equivalent of decimal number.

```
SQL> CREATE OR REPLACE FUNCTION decimal_to_binary (dec_num IN NUMBER)
  2  RETURN VARCHAR2 IS
  3      binary_result VARCHAR2(100) := '';
  4      num NUMBER := dec_num;
  5      remainder NUMBER;
  6  BEGIN
  7      IF num = 0 THEN
  8          RETURN '0';
  9      END IF;
 10
 11      WHILE num > 0 LOOP
 12          remainder := MOD(num, 2);
 13          binary_result := remainder || binary_result;
 14          num := TRUNC(num / 2);
 15      END LOOP;
 16
 17      RETURN binary_result;
 18  END;
 19  /

Function created.

SQL> SELECT decimal_to_binary(10) FROM dual;

DECIMAL_TO_BINARY(10)
--------------------------------------------------------------------------------
1010

SQL>
```

8. Write and execute PL/SQL procedure to transfer fund from one account to another.

```
SQL> CREATE TABLE bank_account (
  2      account_no NUMBER PRIMARY KEY,
  3      account_holder VARCHAR2(100),
  4      balance NUMBER CHECK (balance >= 0)
  5  );

Table created.

SQL> INSERT INTO bank_account VALUES (101, 'Alice', 5000);

1 row created.

SQL> INSERT INTO bank_account VALUES (102, 'Bob', 3000);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> CREATE OR REPLACE PROCEDURE transfer_funds (
  2      sender_acct IN NUMBER,
  3      receiver_acct IN NUMBER,
  4      transfer_amount IN NUMBER
  5  ) AS
  6      sender_balance NUMBER;
  7  BEGIN
  8      SELECT balance INTO sender_balance FROM bank_account WHERE account_no = sender_acct;
  9
 10      IF sender_balance < transfer_amount THEN
 11          RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in sender account');
 12      END IF;
 13
 14      UPDATE bank_account SET balance = balance - transfer_amount WHERE account_no = sender_acct;
 15      UPDATE bank_account SET balance = balance + transfer_amount WHERE account_no = receiver_acct;
 16
 17      COMMIT;
 18
 19      DBMS_OUTPUT.PUT_LINE('Transaction successful! ' || transfer_amount || ' transferred from ' || sender_acct || ' to ' || receiver_acct);
 20  EXCEPTION
 21      WHEN NO_DATA_FOUND THEN
 22          RAISE_APPLICATION_ERROR(-20002, 'One or both accounts do not exist');
 23      WHEN OTHERS THEN
 24          ROLLBACK;
```

```
 20  EXCEPTION
 21      WHEN NO_DATA_FOUND THEN
 22          RAISE_APPLICATION_ERROR(-20002, 'One or both accounts do not exist');
 23      WHEN OTHERS THEN
 24          ROLLBACK;
 25          RAISE_APPLICATION_ERROR(-20003, 'Transaction failed due to an unexpected error');
 26  END;
 27  /

Procedure created.

SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
  2      transfer_funds(101, 102, 1000);
  3  END;
  4  /
Transaction successful! 1000 transferred from 101 to 102

PL/SQL procedure successfully completed.

SQL> SELECT * FROM bank_account;

ACCOUNT_NO
----------
ACCOUNT_HOLDER
--------------------------------------------------------------------------------
   BALANCE
----------
       101
Alice
      4000

       102
Bob
      4000

ACCOUNT_NO
----------
ACCOUNT_HOLDER
--------------------------------------------------------------------------------
   BALANCE
----------
```

9.Write and execute triggers using PL/SQL.

```
SQL*Plus: Release 11.2.0.4.0 Production on Mon Apr 14 14:47:48 2025

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: scott
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> -- Step 1: Drop tables if they already exist (optional, safe for re-run)
SQL> BEGIN
  2      EXECUTE IMMEDIATE 'DROP TABLE emp_audit';
  3  EXCEPTION
  4      WHEN OTHERS THEN NULL;
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
  2      EXECUTE IMMEDIATE 'DROP TABLE employees';
  3  EXCEPTION
  4      WHEN OTHERS THEN NULL;
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> -- Step 2: Create main table
SQL> CREATE TABLE employees (
  2      emp_id NUMBER PRIMARY KEY,
  3      emp_name VARCHAR2(100),
  4      emp_salary NUMBER
  5  );

Table created.

SQL>
SQL> -- Step 3: Create audit table
SQL> CREATE TABLE emp_audit (
  2      emp_id NUMBER,
  3      emp_name VARCHAR2(100),
  4      action_date DATE,
  5      action_type VARCHAR2(20)
  6  );

Table created.

SQL>
SQL> -- Step 4: Create AFTER INSERT trigger
SQL> CREATE OR REPLACE TRIGGER trg_emp_after_insert
  2  AFTER INSERT ON employees
  3  FOR EACH ROW
  4  BEGIN
  5      INSERT INTO emp_audit (emp_id, emp_name, action_date, action_type)
  6      VALUES (:NEW.emp_id, :NEW.emp_name, SYSDATE, 'INSERT');
  7  END;
  8  /

Trigger created.
```

```
SQL> -- Step 5: Create BEFORE UPDATE trigger
SQL> CREATE OR REPLACE TRIGGER trg_emp_before_update
  2  BEFORE UPDATE ON employees
  3  FOR EACH ROW
  4  BEGIN
  5      INSERT INTO emp_audit (emp_id, emp_name, action_date, action_type)
  6      VALUES (:OLD.emp_id, :OLD.emp_name, SYSDATE, 'UPDATE');
  7  END;
  8  /

Trigger created.

SQL>
SQL> -- Step 6: Create BEFORE DELETE trigger
SQL> CREATE OR REPLACE TRIGGER trg_emp_before_delete
  2  BEFORE DELETE ON employees
  3  FOR EACH ROW
  4  BEGIN
  5      INSERT INTO emp_audit (emp_id, emp_name, action_date, action_type)
  6      VALUES (:OLD.emp_id, :OLD.emp_name, SYSDATE, 'DELETE');
  7  END;
  8  /

Trigger created.

SQL>
SQL> -- Step 7: Insert a record
SQL> INSERT INTO employees (emp_id, emp_name, emp_salary)
  2  VALUES (101, 'Aarav', 55000);

1 row created.
```

```
SQL> -- Step 8: Update the record
SQL> UPDATE employees
  2  SET emp_salary = 60000
  3  WHERE emp_id = 101;

1 row updated.

SQL>
SQL> -- Step 9: Delete the record
SQL> DELETE FROM employees
  2  WHERE emp_id = 101;

1 row deleted.

SQL>
SQL> -- Step 10: View the audit log
SQL> SELECT * FROM emp_audit;

    EMP_ID
----------
EMP_NAME
----------------------------------------
ACTION_DA ACTION_TYPE
--------- --------------------
       101
Aarav
14-APR-25 INSERT

       101
Aarav
14-APR-25 UPDATE
```

```
SQL> -- Step 10: View the audit log
SQL> SELECT * FROM emp_audit;

    EMP_ID
----------
EMP_NAME
----------------------------------------
ACTION_DA ACTION_TYPE
--------- --------------------
       101
Aarav
14-APR-25 INSERT

       101
Aarav
14-APR-25 UPDATE

    EMP_ID
----------
EMP_NAME
----------------------------------------
ACTION_DA ACTION_TYPE
--------- --------------------

       101
Aarav
14-APR-25 DELETE


SQL>
```

10. Create and perform database operations using ODBC.

```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>cd C:\Users\LENOVO\Desktop\odbc codes

C:\Users\LENOVO\Desktop\odbc codes>python import.py
Table 'Candidates' created successfully.
Data inserted successfully.

Data in Candidates:
(Decimal('1'), 'Shiv', Decimal('25'))
(Decimal('2'), 'Ram', Decimal('30'))

Data updated successfully.

Record deleted successfully.

Connection closed successfully.

C:\Users\LENOVO\Desktop\odbc codes>
```