CREATE TABLE Users (

UserID INT PRIMARY KEY,

UserName VARCHAR(50)

);

CREATE TABLE Bikes (

BikeID INT PRIMARY KEY,

BikeName VARCHAR(50),

Status VARCHAR(10) DEFAULT 'Available',

DailyRate NUMBER(10,2)

);

CREATE TABLE Rentals (

RentalID INT PRIMARY KEY,

UserID INT,

BikeID INT,

RentalDate DATE,

ReturnDate DATE,

TotalCost DECIMAL(10,2),

FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (BikeID) REFERENCES Bikes(BikeID)

);

--SEQUENCE TO GENRATE USER ID

CREATE SEQUENCE USER\_ID\_SEQ

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

--PL/SQL BLOCK FOR USER REGISTRATION

DECLARE

UID NUMBER;

UName VARCHAR(50);

BEGIN

UName:=:UName;

SELECT USER\_ID\_SEQ.NEXTVAL INTO UID FROM DUAL;

INSERT INTO Users VALUES(UID,UName);

DBMS\_OUTPUT.PUT\_LINE('YOUR USER ID IS '||UID);

COMMIT;

END;

--SEQUENCE TO GENRATE RENTAL ID

CREATE SEQUENCE RENT\_ID\_SEQ

START WITH 1001

INCREMENT BY 1

NOCACHE

NOCYCLE;

--TRIGGER FOR RENTAL ID

CREATE OR REPLACE TRIGGER RENTID\_TRIGGER

BEFORE INSERT ON Rentals

FOR EACH ROW

BEGIN

SELECT RENT\_ID\_SEQ.NEXTVAL INTO :NEW.RentalID FROM DUAL;

END;

--PACKAGE RENT SPECIFICATION

CREATE OR REPLACE PACKAGE RENT

IS

PROCEDURE RentBike(p\_user\_id INT,p\_bike\_id INT,p\_rental\_days INT);

PROCEDURE ReturnBike(p\_bike\_name VARCHAR2,p\_user\_name VARCHAR2);

FUNCTION CalculateRentalCost(p\_rental\_days INT,p\_daily\_rate DECIMAL) RETURN DECIMAL;

END RENT;

--PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY RENT

IS

-- Function for rental cost calculation

FUNCTION CalculateRentalCost(p\_rental\_days INT,p\_daily\_rate DECIMAL) RETURN DECIMAL

IS

v\_cost DECIMAL(10,2);

BEGIN

v\_cost := p\_rental\_days \* p\_daily\_rate;

RETURN v\_cost;

END;

-- Procedure for renting a bike

PROCEDURE RentBike(p\_user\_id INT,p\_bike\_id INT,p\_rental\_days INT)

IS

v\_daily\_rate DECIMAL(10,2);

v\_total\_cost DECIMAL(10,2);

v\_bike\_id NUMBER;

v\_status VARCHAR(10);

BEGIN

SELECT BikeID INTO v\_bike\_id from Bikes WHERE BikeID = p\_bike\_id;

SELECT Status INTO v\_status from Bikes WHERE BikeID = p\_bike\_id;

IF v\_bike\_id = p\_bike\_id AND v\_status = 'Occupied' THEN

RAISE\_APPLICATION\_ERROR(-20008,'BIKE IS ALREADY OCCUPIED');

ELSE

-- Get daily rate from Bike table

SELECT DailyRate INTO v\_daily\_rate FROM Bikes WHERE BikeID = p\_bike\_id;

-- Calculate total cost using the function

v\_total\_cost := CalculateRentalCost(p\_rental\_days, v\_daily\_rate);

dbms\_output.put\_line('YOUR TOTAL COST IS'||v\_total\_cost);

-- Insert rental record

INSERT INTO Rentals (UserID, BikeID, RentalDate, ReturnDate, TotalCost)

VALUES (p\_user\_id, p\_bike\_id, SYSDATE, NULL, v\_total\_cost);

-- Update bike status

UPDATE Bikes SET status = 'Occupied' WHERE BikeID = p\_bike\_id;

COMMIT;

END IF;

END;

--Procedure to return a bike

PROCEDURE ReturnBike (p\_bike\_name VARCHAR2,p\_user\_name VARCHAR2)

IS

v\_return\_date DATE := SYSDATE;

v\_total\_cost DECIMAL(10,2);

v\_rental\_days INT;

v\_daily\_rate DECIMAL(10,2);

BEGIN

--UPDATE RENTURNDATE FOR THE RENTAL

UPDATE Rentals r set returndate=v\_return\_date WHERE

r.bikeid IN (SELECT BikeID FROM Bikes WHERE Bikename = p\_bike\_name)

AND r.userid IN (SELECT userid FROM Users WHERE username = p\_user\_name)

AND returndate IS NULL;

COMMIT;

END;

END RENT;

--TRIGGER TO UPDATE BIKE STATUS

CREATE OR REPLACE TRIGGER UpdateBikeStatus

AFTER UPDATE ON Rentals

FOR EACH ROW

BEGIN

--CHECK IF THE RENTAL WAS MARKED AS RETURNED

IF :NEW.ReturnDate IS NOT NULL

AND :OLD.ReturnDate IS NULL THEN

UPDATE Bikes SET Status = 'Available' WHERE BikeID = :NEW.BikeID;

END IF;

END;

--PL/SQL BLOCK TO RENT BIKE TO USER

DECLARE

UName VARCHAR(50);

UBike VARCHAR(50);

udays NUMBER;

uid NUMBER;

bid NUMBER;

BEGIN

UName:=:UName;

UBike:=:UBike;

udays:=:udays;

SELECT UserID into uid FROM Users where username = UName;

SELECT BikeID into bid FROM Bikes where bikename=UBike ;

RENT.rentbike(uid,bid,udays);

END;

--PL/SQL BLOCK TO RETURN BIKE FROM USER

DECLARE

uname VARCHAR(50);

ubike VARCHAR(50);

BEGIN

uname:=:uname;

ubike:=:ubike;

RENT.returnbike(ubike,uname);

END;