**Project Database Design**

**Hotel – Room Service**

***Team:***

***Patrut Anita – Alexandra***

***Iorga Adina***

***Petrisor Paul - Andrei***

**TABLES**

* **The Client Table**

CREATE TABLE Client(

ID INTEGER PRIMARY KEY,

Name VARCHAR2(80) NOT NULL,

GENDER CHAR NOT NULL

);

* **The Room Table**

CREATE TABLE Room(

ID INTEGER PRIMARY KEY,

Working\_day\_price NUMBER(7,2) NOT NULL,

Weekend\_price NUMBER(7,2) NOT NULL,

No\_of\_places INTEGER NOT NULL

);

* **The Reservation Table**

CREATE TABLE Reservation(

Client\_ID INTEGER,

Room\_ID INTEGER,

Start\_Date Date NOT NULL,

End\_Date Date NOT NULL,

Fee NUMBER(7,2) NOT NULL,

CONSTRAINT fk\_client FOREIGN KEY (Client\_ID) REFERENCES Client(ID),

CONSTRAINT fk\_room FOREIGN KEY (Room\_ID) REFERENCES Room(ID)

);

**REQUIREMENTS**

1. **Write an SQL file to create the database structure and define all necessary Key Constraints. Add to this file also some INSERT statements to add at least ten clients and ten hotel rooms.**

INSERT ALL INTO Client VALUES (1,'Tori Koch','F')

INTO Client VALUES (2,'Dane Mclean','M')

INTO Client VALUES (3,'Roselyn Arellano','F')

INTO Client VALUES (4,'Pamela Nash','F')

INTO Client VALUES (5,'Bo Dyer','M')

INTO Client VALUES (6,'Tamia Bush','F')

INTO Client VALUES (7,'Gracelyn Barry','F')

INTO Client VALUES (8,'Averi Lutz','M')

INTO Client VALUES (9,'Hayden Green','M')

INTO Client VALUES (10,'Camilla Dennis','F')

INTO Client VALUES (11,'Steve Grant','M')

INTO Client VALUES (12,'Porter Warner','M')

INTO Client VALUES (13,'Olive Hurley','M')

INTO Client VALUES (14,'Yareli Bautista','F')

INTO Client VALUES (15,'Isabela Douglas','F')

SELECT \* FROM dual;

INSERT ALL INTO Room VALUES (100,120,140,2)

INTO Room VALUES (101,120,140,2)

INTO Room VALUES (102,100,120,2)

INTO Room VALUES (103,100,120,2)

INTO Room VALUES (104,100,120,2)

INTO Room VALUES (105,120,140,2)

INTO Room VALUES (106,160,180,1)

INTO Room VALUES (107,160,180,1)

INTO Room VALUES (108,160,180,1)

INTO Room VALUES (109,100,120,1)

INTO Room VALUES (110,180,200,2)

INTO Room VALUES (111,180,200,2)

SELECT \* FROM dual;

1. **Write a stored procedure for booking a room. The current date will be used as start date, while the rest of necessary information is passed by parameters (Client\_ID, Room\_ID. No\_of\_days). The No\_of\_days will be used to compute the agreed End\_date. The procedure has to compute the fee based on Working\_days\_price and Weekend\_price from the Room table.**

CREATE OR REPLACE PROCEDURE Book\_a\_room(Client\_ID IN Client.ID%Type,

Room\_ID IN Room.ID%Type,

No\_of\_Days IN INTEGER) IS

Start\_Date Date;

End\_Date Date;

No\_of\_Working\_Days INTEGER;

No\_of\_Weekend\_Days INTEGER;

k INTEGER;

Fee NUMBER(7,2);

Working\_price NUMBER(7,2);

Weekend\_price NUMBER(7,2);

BEGIN

SELECT SYSDATE INTO Start\_Date FROM dual;

End\_Date := Start\_Date + No\_of\_Days - 1;

No\_of\_Working\_Days := 0;

No\_of\_Weekend\_Days := 0;

Fee := 0;

for k IN 0..No\_of\_Days-1 LOOP

IF TO\_CHAR(Start\_Date+k, 'd') = 6 OR TO\_CHAR(Start\_Date+k, 'd') = 7 THEN

No\_of\_Weekend\_Days := No\_of\_Weekend\_Days + 1;

ELSE

No\_of\_Working\_Days := No\_of\_Working\_Days + 1;

END IF;

END LOOP;

SELECT Working\_day\_price,Weekend\_price INTO Working\_price,Weekend\_price FROM Room WHERE ID = Room\_ID;

Fee := Working\_price\*No\_of\_Working\_Days + Weekend\_price\*No\_of\_Weekend\_Days;

INSERT INTO Reservation VALUES(Client\_ID,Room\_ID,Start\_Date,End\_Date,Fee);

DBMS\_output.Put\_line('Room is successfully reserved!');

END;

1. **Add a trigger (lab9) for Reservation table to prohibit a new reservation or a modification applied on an existing one (when the Room\_ID changes) if it exceeds the capacity of the room (No\_of\_places from Room table).**

CREATE OR REPLACE TRIGGER trigger\_3

BEFORE INSERT OR UPDATE ON Reservation

FOR EACH ROW

DECLARE

pragma AUTONOMOUS\_TRANSACTION;

Actual\_No\_Of\_Places INTEGER;

Places\_Reserved INTEGER;

BEGIN

SELECT No\_of\_places INTO Actual\_No\_Of\_Places FROM Room WHERE ID = :NEW.Room\_ID;

SELECT COUNT(\*) INTO Places\_Reserved FROM Reservation WHERE Room\_ID = :NEW.Room\_ID;

IF Places\_Reserved = Actual\_No\_Of\_Places THEN

RAISE\_APPLICATION\_ERROR( -20001,

'There is no more space in the room!' );

END IF;

commit;

END

1. **Write a procedure for accommodation ending. It will automatically inject the current date into the actual End\_date field, and compute the corresponding fee based on Working\_days\_price and Weekend\_price from the Room table. If the End\_date differs from agreed End\_date, a penalty of 10% should be applied.**

CREATE OR REPLACE PROCEDURE Accomodation\_Ending(C\_ID IN Client.ID%Type, R\_ID IN Room.ID%Type) IS

TYPE Reserv IS RECORD

(Client\_ID INTEGER,

Room\_ID INTEGER,

Start\_Date Date,

End\_Date Date,

Fee NUMBER(7,2));

R1 Reserv;

Current\_Date Date;

Extra\_Days INTEGER;

d DATE;

Extra\_working\_days\_no NUMBER(7,2);

Extra\_weekend\_days\_no NUMBER(7,2);

Working\_day\_price NUMBER(7,2);

Weekend\_price NUMBER(7,2);

Extra\_pay NUMBER(7,2);

BEGIN

SELECT SYSDATE INTO Current\_Date FROM dual;

SELECT \* INTO R1 FROM Reservation WHERE (Client\_ID = C\_ID AND Room\_ID = R\_ID);

IF Current\_Date > R1.End\_Date+1 THEN

SELECT Working\_day\_price,Weekend\_price INTO Working\_day\_price,Weekend\_price FROM Room WHERE ID = R1.Room\_ID;

Extra\_working\_days\_no := 0;

Extra\_weekend\_days\_no := 0;

Extra\_pay :=0;

d := R1.End\_Date+1;

Extra\_days := Current\_Date - R1.End\_Date-1;

FOR k IN 0..Extra\_days LOOP

IF TO\_CHAR(d, 'd') = 1 OR TO\_CHAR(d, 'd') = 7 THEN

Extra\_weekend\_days\_no := Extra\_weekend\_days\_no + 1;

ELSE

Extra\_working\_days\_no := Extra\_working\_days\_no + 1;

END IF;

d := d+1;

END LOOP;

Extra\_pay := Extra\_working\_days\_no\*Working\_day\_price + Extra\_weekend\_days\_no\*Weekend\_price;

UPDATE Reservation SET

End\_Date = Current\_Date,

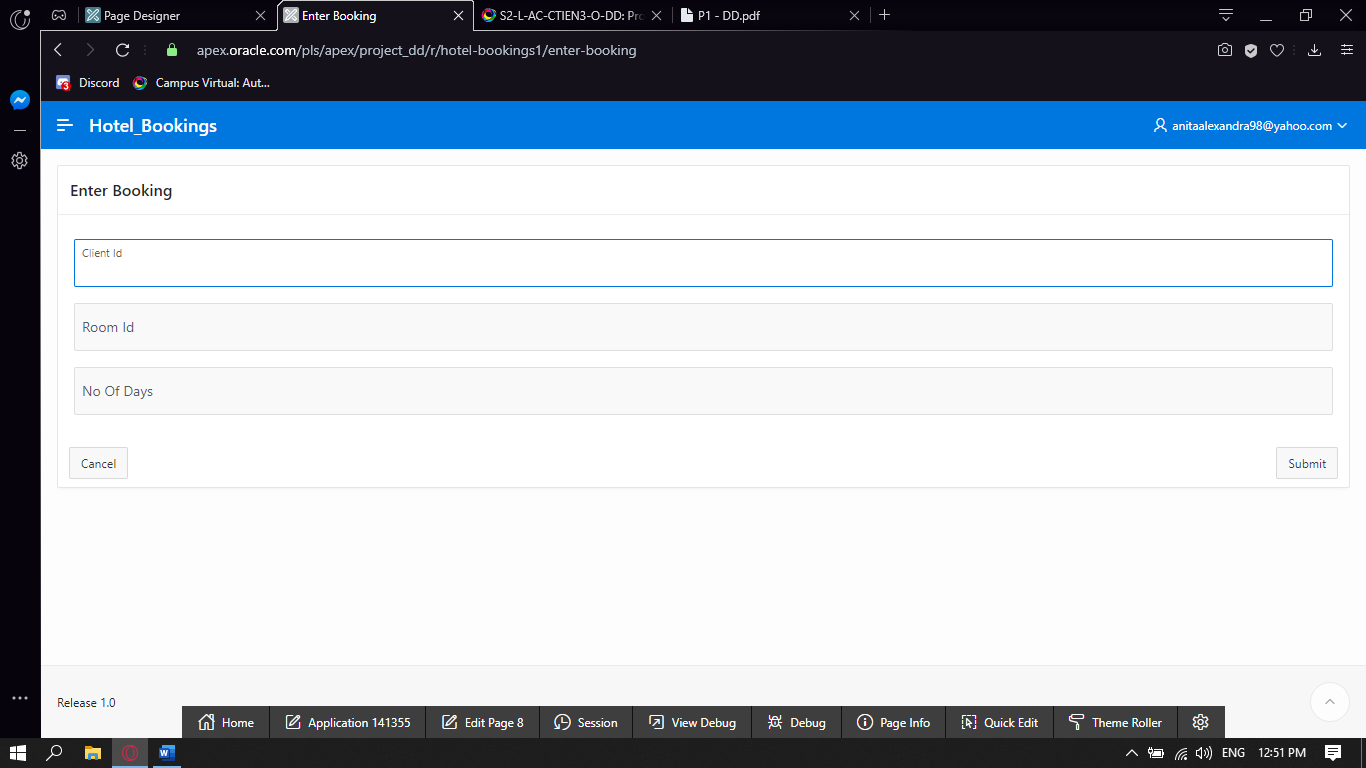
fee = fee + Extra\_pay + 0.1\*Extra\_pay

WHERE Room\_ID = R1.Room\_ID AND Client\_ID = R1.Client\_ID;

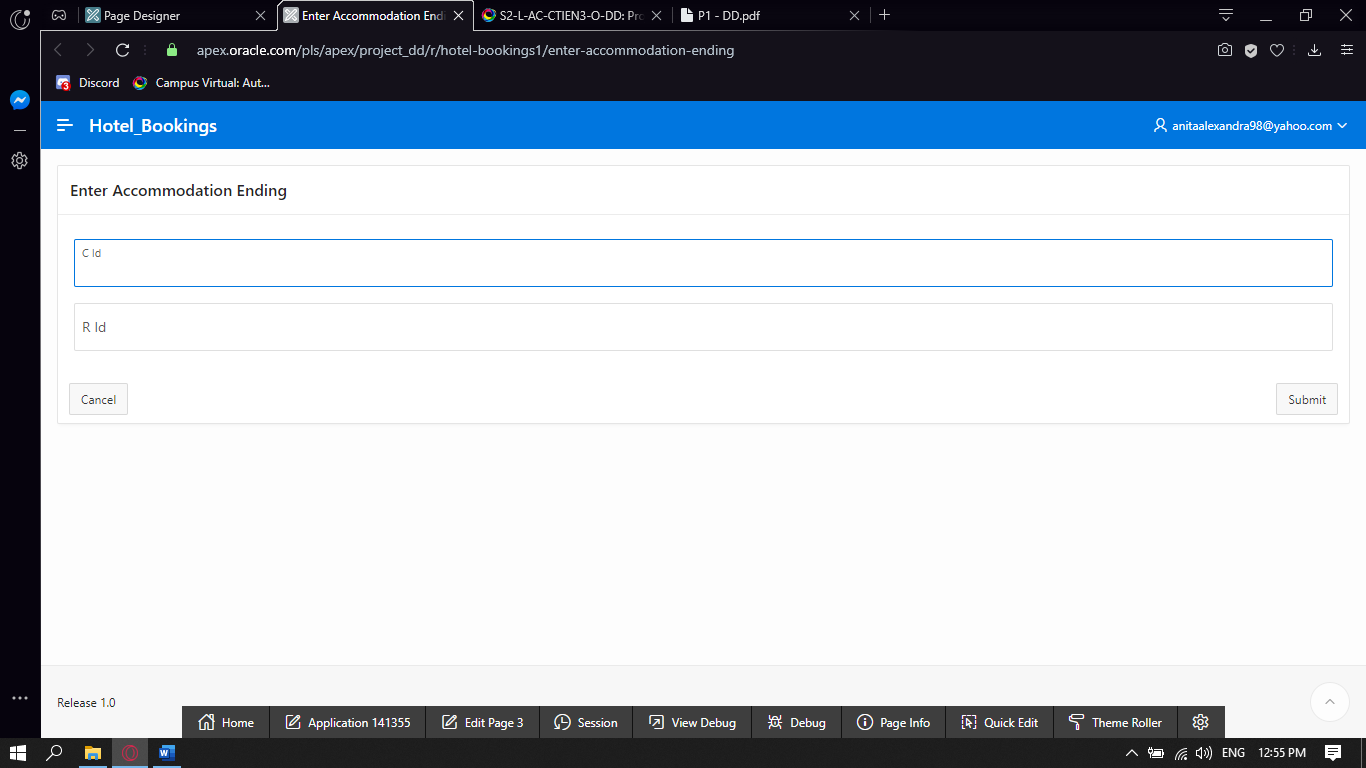
END IF;

END ;

1. **Add an APEX form to input the information necessary to make a reservation. This form will use this information to call the procedure for room booking.**

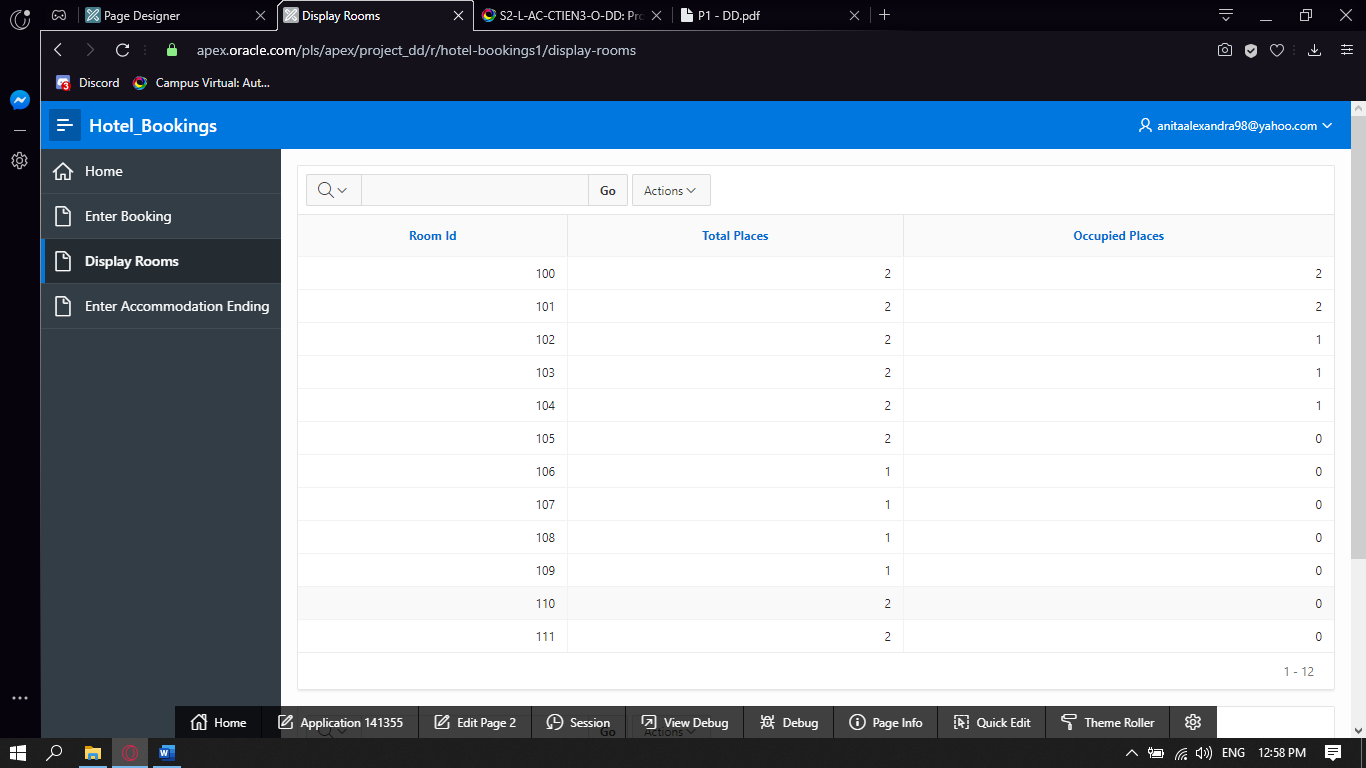
****

1. **Add an APEX form to input the information necessary to ending a reservation. This form will use this information to call the procedure for accommodation ending.**

****

1. **Add an APEX report to list all rooms with information on actual occupancy of each room. The form will display also the number of overall free places.**

SELECT ID AS Room\_ID,No\_of\_Places AS Total\_Places,(select COUNT(\*) FROM reservation Res WHERE Res.Room\_ID=R.ID) AS Occupied\_Places from room R;



SELECT

(SELECT SUM(no\_of\_places)AS TOTAL\_PLACES FROM ROOM) -

(SELECT COUNT(\*) as TOTAL\_RESERVED FROM RESERVATION)

AS Free\_places

FROM DUAL;

