Subqueries and Merge Statements

Ayushi Tiwari

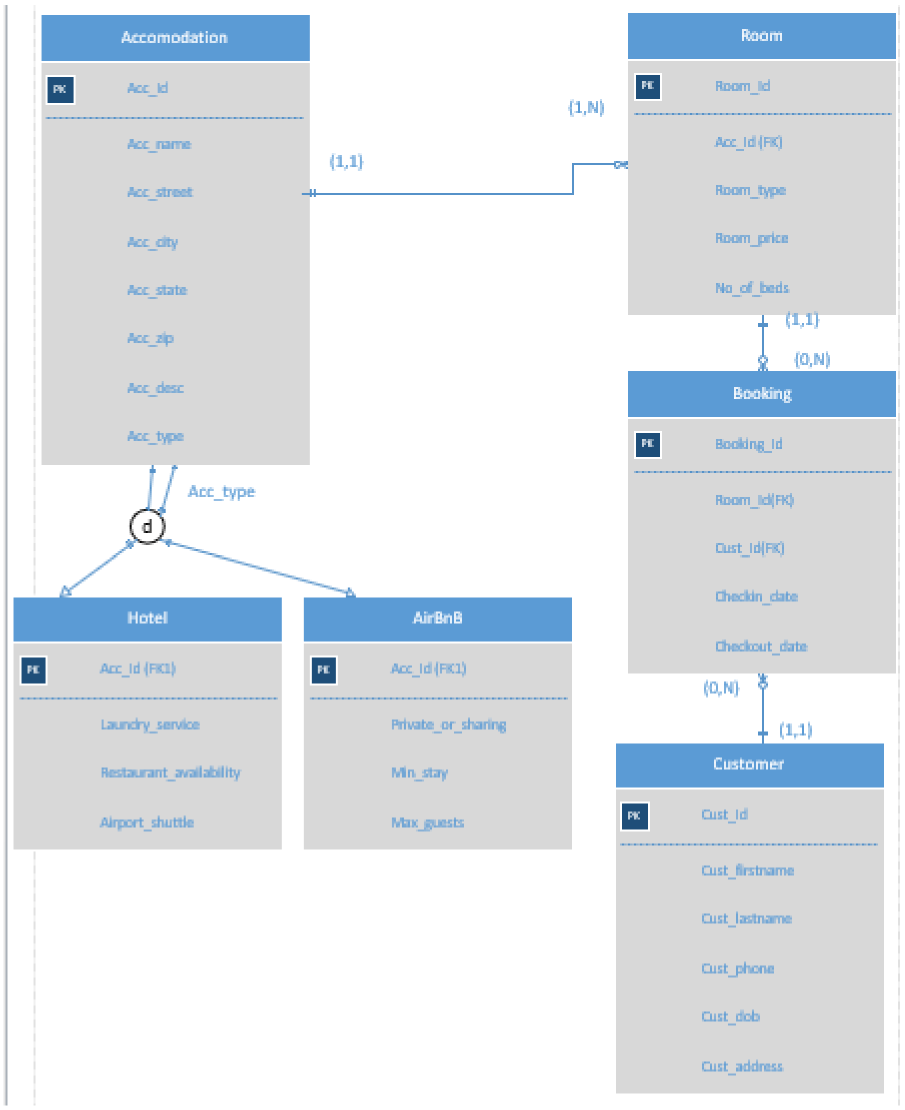
George Mason University

Author Note

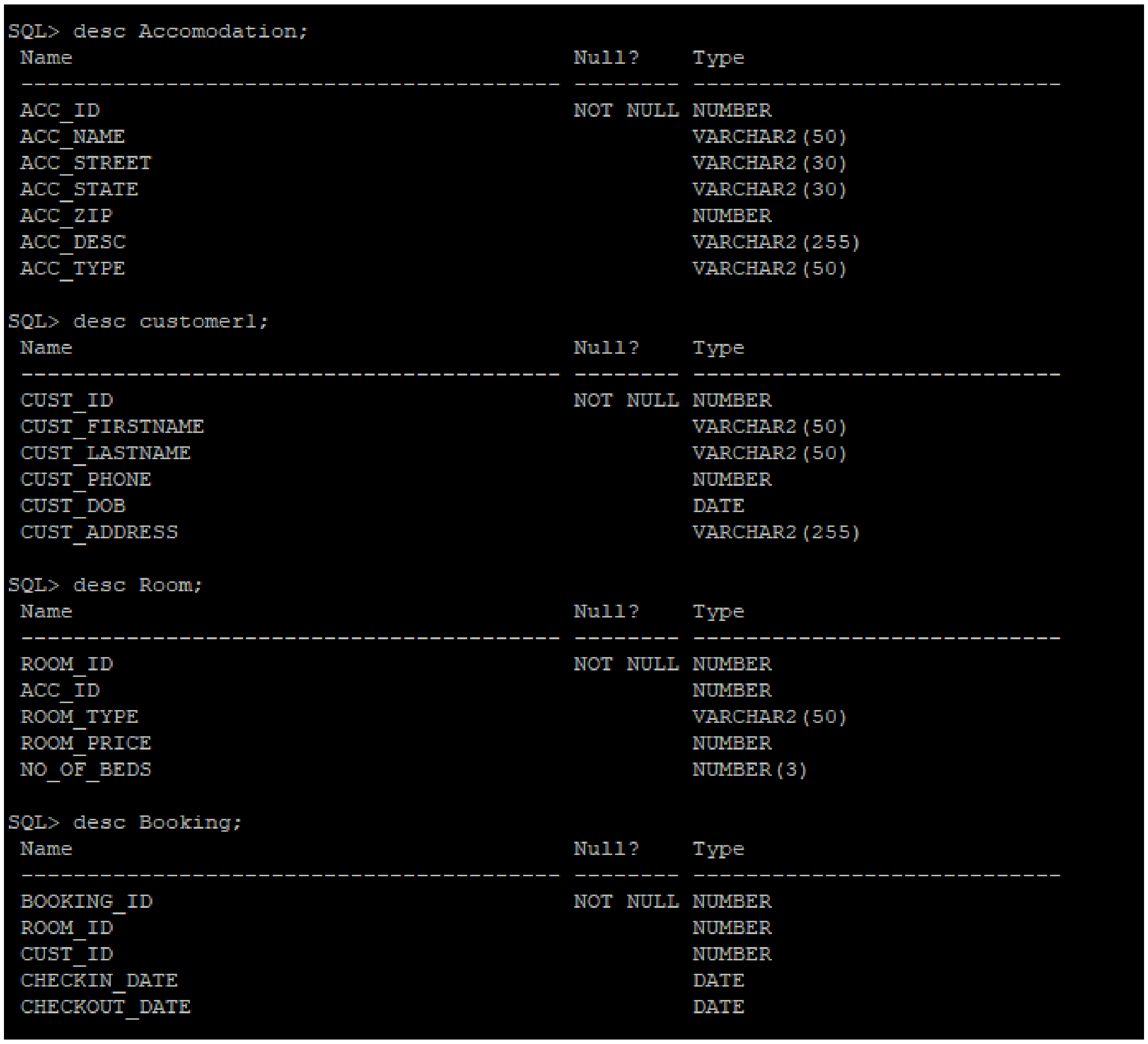
Ayushi Tiwari, Data Analytics Engineering, Volgenau School of Engineering

Contact: atiwari4@masonlive.gmu.edu

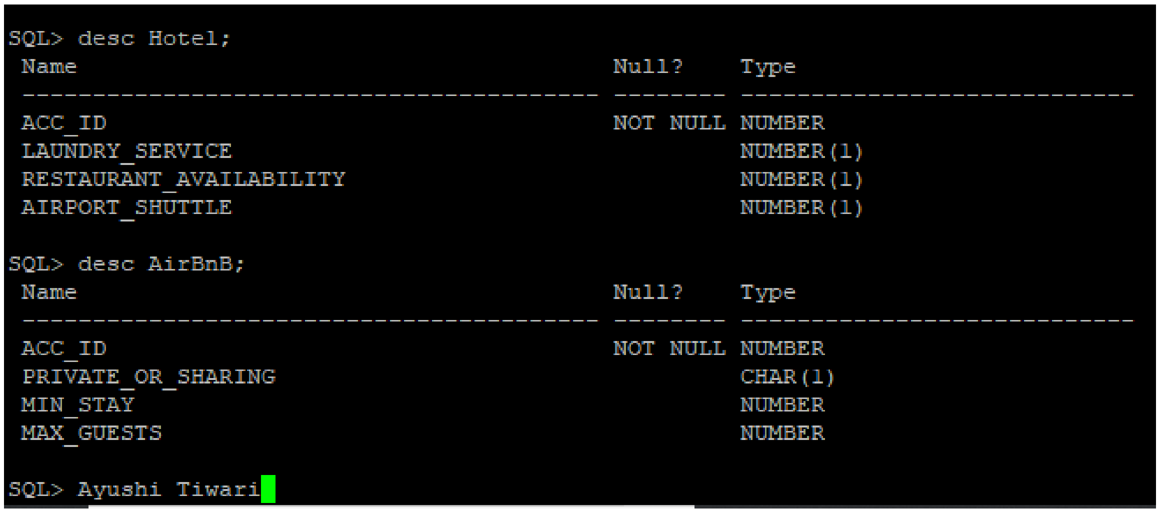
ERD FOR ABC TRAVEL COMPANY


Accomodation
PK Acc_Id
Acc_name
Acc_street
Acc_city
Acc_state
Acc_zip
Acc_desc
Acc_type
(1,1)
(1,N)
Room
PK Room_Id
Acc_Id (FK)
Room_type
Room_price
No_of_beds
(1,1)
(0,N)
Booking
PK Booking_Id
Room_Id(FK)
Cust_Id(FK)
Checkin_date
Checkout_date
(0,N)
(1,1)
Customer
PK Cust_Id
Cust_firstname
Cust_lastname
Cust_phone
Cust_dob
Cust_address
Acc_type
d
Hotel
PK Acc_Id (FK1)
Laundry_service
Restaurant_availability
Airport_shuttle
AirBnB
PK Acc_Id (FK1)
Private_or_sharing
Min_stay
Max_guests

A look at the Table structures in our database:



```
SQL> desc Accomodation;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACC_ID                                    NOT NULL NUMBER
 ACC_NAME                                           VARCHAR2(50)
 ACC_STREET                                         VARCHAR2(30)
 ACC_STATE                                          VARCHAR2(30)
 ACC_ZIP                                            NUMBER
 ACC_DESC                                           VARCHAR2(255)
 ACC_TYPE                                           VARCHAR2(50)

SQL> desc customer1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUST_ID                                   NOT NULL NUMBER
 CUST_FIRSTNAME                                     VARCHAR2(50)
 CUST_LASTNAME                                      VARCHAR2(50)
 CUST_PHONE                                         NUMBER
 CUST_DOB                                           DATE
 CUST_ADDRESS                                       VARCHAR2(255)

SQL> desc Room;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ROOM_ID                                   NOT NULL NUMBER
 ACC_ID                                             NUMBER
 ROOM_TYPE                                          VARCHAR2(50)
 ROOM_PRICE                                         NUMBER
 NO_OF_BEDS                                         NUMBER(3)

SQL> desc Booking;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOKING_ID                                NOT NULL NUMBER
 ROOM_ID                                            NUMBER
 CUST_ID                                            NUMBER
 CHECKIN_DATE                                       DATE
 CHECKOUT_DATE                                      DATE
```

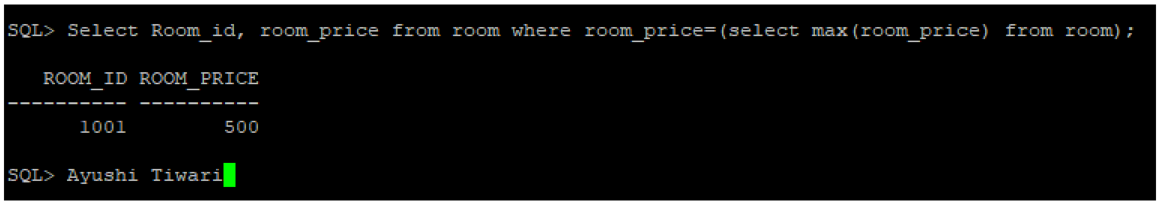


```
SQL> desc Hotel;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACC_ID                                    NOT NULL NUMBER
 LAUNDRY_SERVICE                                    NUMBER(1)
 RESTAURANT_AVAILABILITY                            NUMBER(1)
 AIRPORT_SHUTTLE                                    NUMBER(1)

SQL> desc AirBnB;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACC_ID                                    NOT NULL NUMBER
 PRIVATE_OR_SHARING                                 CHAR(1)
 MIN_STAY                                           NUMBER
 MAX_GUESTS                                         NUMBER

SQL> Ayushi Tiwari
```

For the database you designed and created in previous HWs, complete the following problems:

1. **Write an SQL query that uses a single-row subquery in a WHERE clause. Explain what the query is intended to do.**

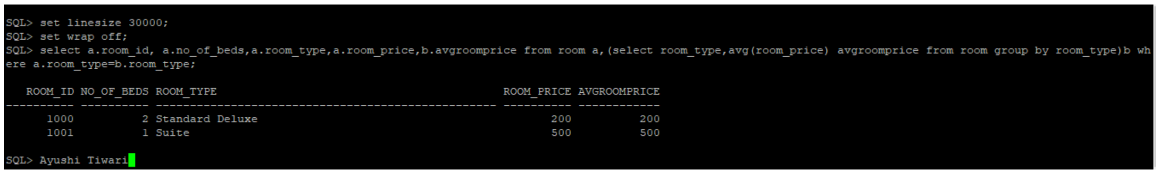
Retrieve room id and price for the costliest room.



```
SQL> Select Room_id, room_price from room where room_price=(select max(room_price) from room);

   ROOM_ID ROOM_PRICE
---------- ----------
      1001        500

SQL> Ayushi Tiwari
```

2. **Write an SQL query that uses a multiple-column subquery in a FROM clause. Explain what the query is intended to do.**

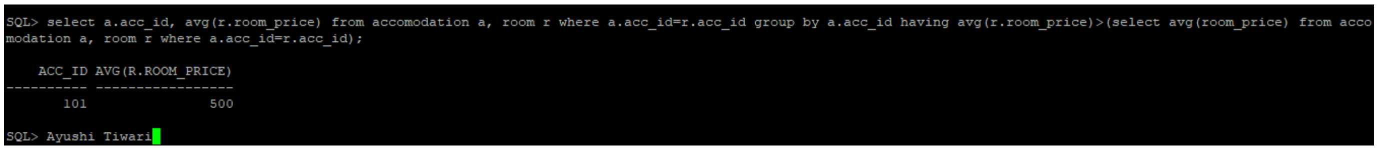
Retrieve room information for rooms along with their average room price based on room type and individual room price.



```
SQL> set linesize 30000;
SQL> set wrap off;
SQL> select a.room_id, a.no_of_beds,a.room_type,a.room_price,b.avgroomprice from room a,(select room_type,avg(room_price) avgroomprice from room group by room_type)b wh
ere a.room_type=b.room_type;

   ROOM_ID NO_OF_BEDS ROOM_TYPE                                          ROOM_PRICE AVGROOMPRICE
---------- ---------- -------------------------------------------------- ---------- ------------
      1000          2 Standard Deluxe                                           200          200
      1001          1 Suite                                                     500          500

SQL> Ayushi Tiwari
```

3. **Write an SQL query that is based on multiple tables and uses a subquery with the GROUP BY statement and HAVING clause. Explain what the query is intended to do.**

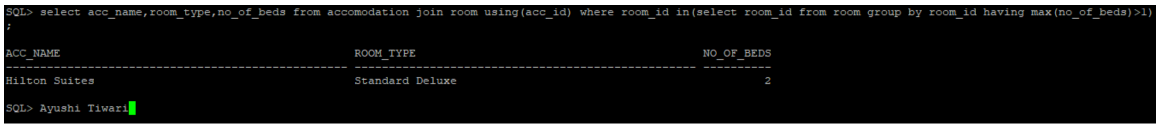
Retrieve accomodation information for those rooms where room price is greater than the average room price of the various type of rooms.



```
SQL> select a.acc_id, avg(r.room_price) from accomodation a, room r where a.acc_id=r.acc_id group by a.acc_id having avg(r.room_price)>(select avg(room_price) from acco
modation a, room r where a.acc_id=r.acc_id);

    ACC_ID AVG(R.ROOM_PRICE)
---------- -----------------
       101               500

SQL> Ayushi Tiwari
```

4. Write an SQL query that is based on multiple tables and uses a multiple-row subquery in a WHERE clause. The subquery will include the GROUP BY statement and another multiple-row subquery in a HAVING clause. Explain what the query is intended to do.

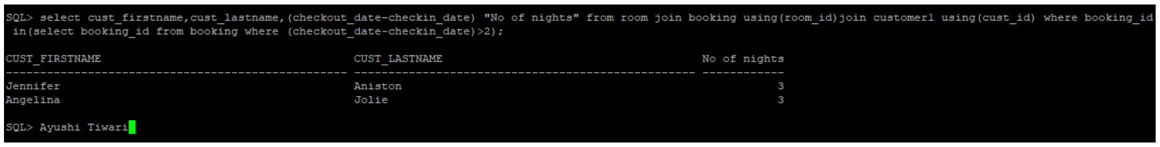
Retrieve the list of all the accommodations that have more than 2 beds.



```
SQL> select acc_name,room_type,no_of_beds from accomodation join room using(acc_id) where room_id in(select room_id from room group by room_id having max(no_of_beds)>1)
;

ACC_NAME                                           ROOM_TYPE                                          NO_OF_BEDS
-------------------------------------------------- -------------------------------------------------- ----------
Hilton Suites                                      Standard Deluxe                                             2

SQL> Ayushi Tiwari
```

5. Write an SQL query that joins three tables and uses any type of a subquery. Explain what the query is intended to do.

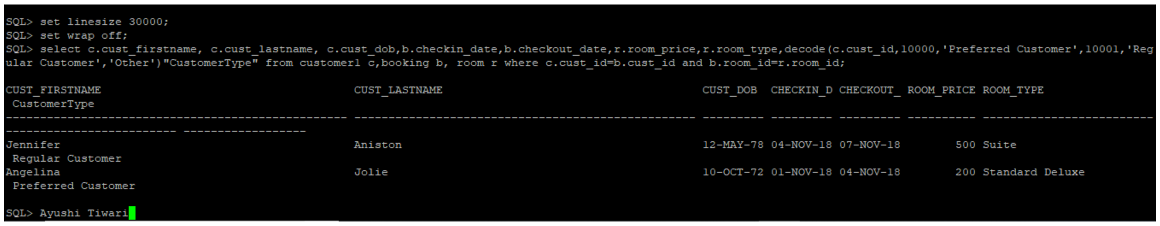
Retrieve customer name and address for those customers who booked a hotel room for more than 2 nights.



```
SQL> select cust_firstname,cust_lastname,(checkout_date-checkin_date) "No of nights" from room join booking using(room_id)join customer1 using(cust_id) where booking_id
 in(select booking_id from booking where (checkout_date-checkin_date)>2);

CUST_FIRSTNAME                                     CUST_LASTNAME                                      No of nights
-------------------------------------------------- -------------------------------------------------- ------------
Jennifer                                           Aniston                                                       3
Angelina                                           Jolie                                                         3

SQL> Ayushi Tiwari
```

6. Write an SQL query that is based on multiple tables and uses the DECODE function. Explain what the query is intended to do.

Assign Preferred and Regular customer labels to customers based on their customer id codes.



```
SQL> set linesize 30000;
SQL> set wrap off;
SQL> select c.cust_firstname, c.cust_lastname, c.cust_dob,b.checkin_date,b.checkout_date,r.room_price,r.room_type,decode(c.cust_id,10000,'Preferred Customer',10001,'Reg
ular Customer','Other')"CustomerType" from customer1 c,booking b, room r where c.cust_id=b.cust_id and b.room_id=r.room_id;

CUST_FIRSTNAME                                     CUST_LASTNAME                                      CUST_DOB  CHECKIN_D CHECKOUT_ ROOM_PRICE ROOM_TYPE
 CustomerType
-------------------------------------------------- -------------------------------------------------- --------- --------- --------- ---------- -------------------------
------------------------- ------------------
Jennifer                                           Aniston                                            12-MAY-78 04-NOV-18 07-NOV-18        500 Suite
 Regular Customer
Angelina                                           Jolie                                              10-OCT-72 01-NOV-18 04-NOV-18        200 Standard Deluxe
 Preferred Customer

SQL> Ayushi Tiwari
```

For Problems 7-10 below, consider the following business scenario:
A university wants to keep track of bonuses given to the professors who mentor (supervise) junior faculty members. Data about all professors is available in the Faculty table (see below). In that table, the f_super column represents a faculty id of a mentor (if any). A separate Bonus table is needed to assign and keep track of bonuses. The Bonus table will have a default bonus of 1000. It will be updated once a year to add new mentors and to update bonuses for the existing ones.

7. Create the Faculty table and populate it with data using the script below:

CREATE TABLE faculty (f_id NUMBER(6), f_last VARCHAR2(30) ,f_first VARCHAR2(30), f_mi CHAR(1), loc_id NUMBER(5), f_phone VARCHAR2(10), f_rank VARCHAR2(9), f_super NUMBER(6), CONSTRAINT faculty_f_id_pk PRIMARY KEY(f_id));

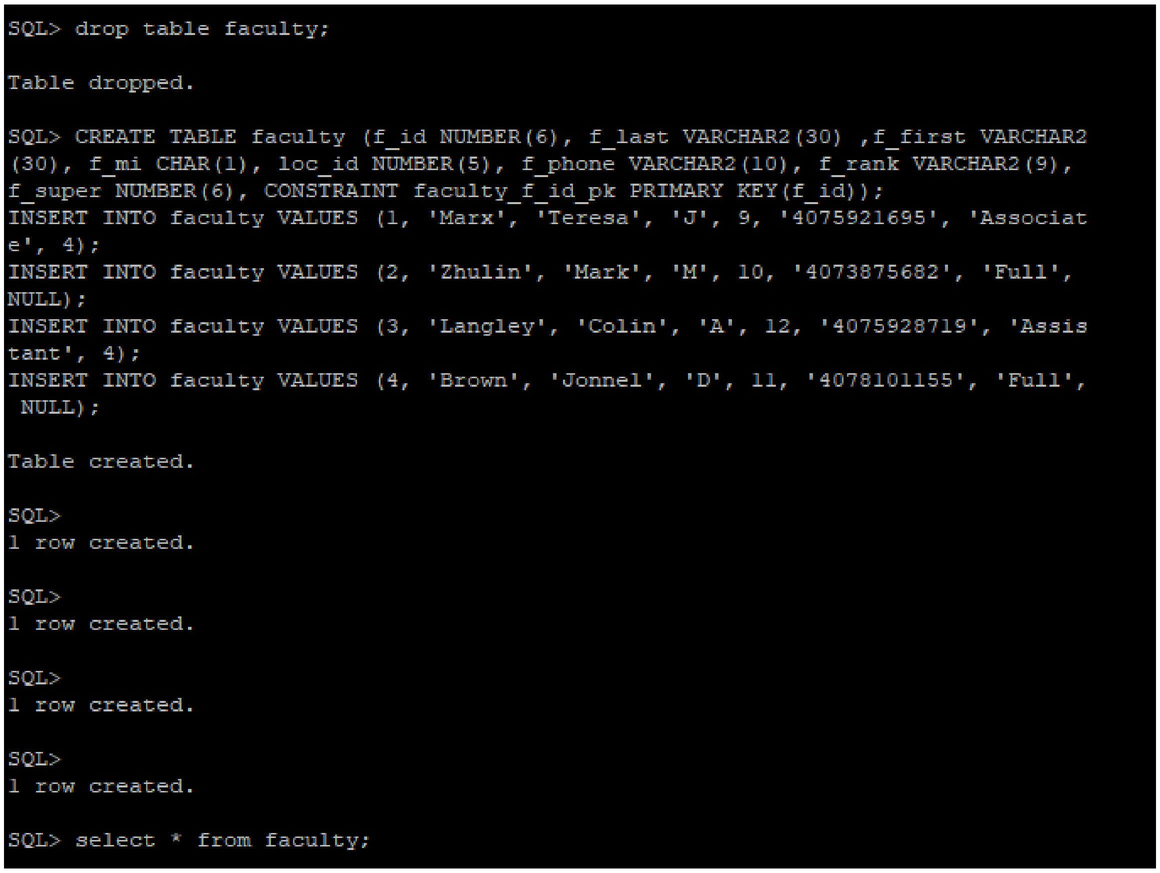
INSERT INTO faculty VALUES (1, 'Marx', 'Teresa', 'J', 9, '4075921695', 'Associate', 4);

INSERT INTO faculty VALUES (2, 'Zhulin', 'Mark', 'M', 10, '4073875682', 'Full', NULL);

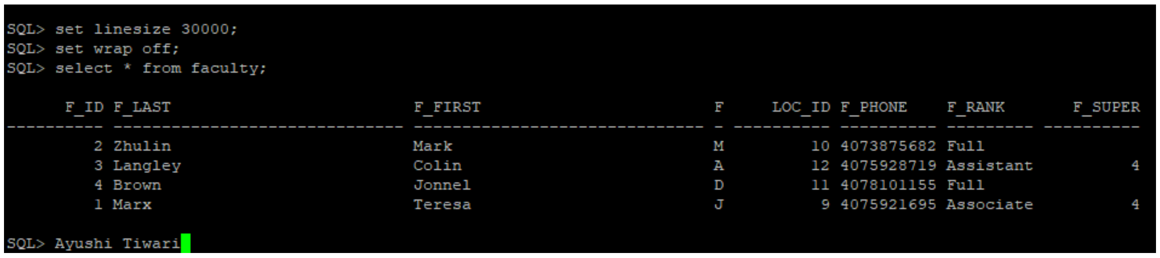
INSERT INTO faculty VALUES (3, 'Langley', 'Colin', 'A', 12, '4075928719', 'Assistant', 4);

INSERT INTO faculty VALUES (4, 'Brown', 'Jonnel', 'D', 11, '4078101155', 'Full', NULL);

Check the result using the *select * from faculty;* command.



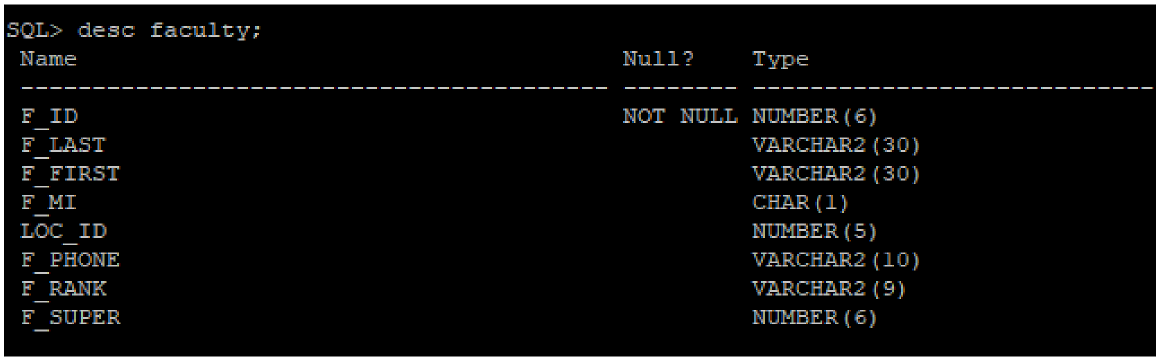
```
SQL> drop table faculty;

Table dropped.

SQL> CREATE TABLE faculty (f_id NUMBER(6), f_last VARCHAR2(30) ,f_first VARCHAR2
(30), f_mi CHAR(1), loc_id NUMBER(5), f_phone VARCHAR2(10), f_rank VARCHAR2(9),
f_super NUMBER(6), CONSTRAINT faculty_f_id_pk PRIMARY KEY(f_id));
INSERT INTO faculty VALUES (1, 'Marx', 'Teresa', 'J', 9, '4075921695', 'Associat
e', 4);
INSERT INTO faculty VALUES (2, 'Zhulin', 'Mark', 'M', 10, '4073875682', 'Full',
NULL);
INSERT INTO faculty VALUES (3, 'Langley', 'Colin', 'A', 12, '4075928719', 'Assis
tant', 4);
INSERT INTO faculty VALUES (4, 'Brown', 'Jonnel', 'D', 11, '4078101155', 'Full',
 NULL);

Table created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> select * from faculty;
```



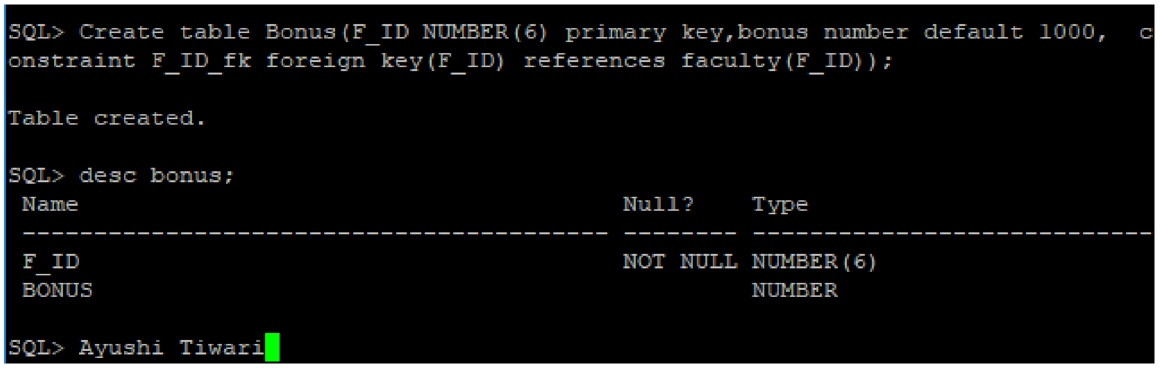
```
SQL> set linesize 30000;
SQL> set wrap off;
SQL> select * from faculty;

      F_ID F_LAST                         F_FIRST                        F     LOC_ID F_PHONE    F_RANK       F_SUPER
---------- ------------------------------ ------------------------------ - ---------- ---------- --------- ----------
         2 Zhulin                         Mark                           M         10 4073875682 Full
         3 Langley                        Colin                          A         12 4075928719 Assistant          4
         4 Brown                          Jonnel                         D         11 4078101155 Full
         1 Marx                           Teresa                         J          9 4075921695 Associate          4

SQL> Ayushi Tiwari
```

8. Create the Bonus table that consists of two columns: f_id (PK) and bonus. For the f_id column, use the same description as in the Faculty table. For the bonus column, use the NUMBER data type and the DEFAULT constraint to set the values for the bonus column to 1000 (bonus amount). Next, use a subquery to copy ids of mentors given in the Faculty table into the Bonus table.

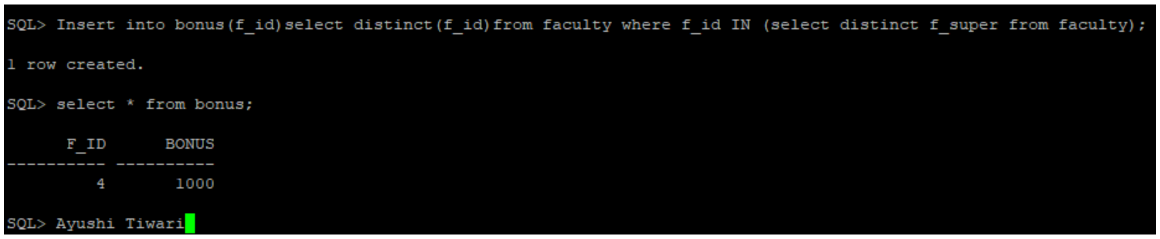
Check the result using the *select * from bonus;* command.



```
SQL> desc faculty;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 F_ID                                      NOT NULL NUMBER(6)
 F_LAST                                             VARCHAR2(30)
 F_FIRST                                            VARCHAR2(30)
 F_MI                                               CHAR(1)
 LOC_ID                                             NUMBER(5)
 F_PHONE                                            VARCHAR2(10)
 F_RANK                                             VARCHAR2(9)
 F_SUPER                                            NUMBER(6)
```



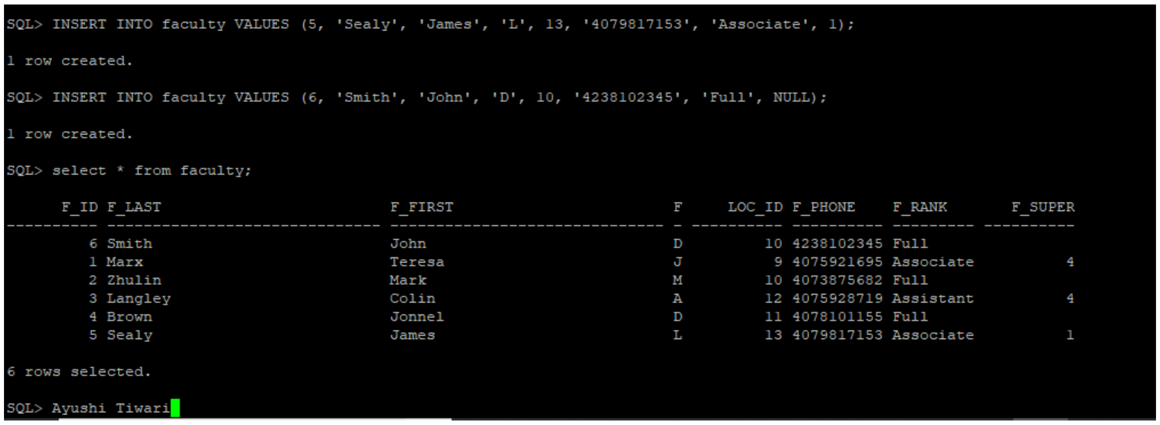
```
SQL> Create table Bonus(F_ID NUMBER(6) primary key,bonus number default 1000,  c
onstraint F_ID_fk foreign key(F_ID) references faculty(F_ID));

Table created.

SQL> desc bonus;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 F_ID                                      NOT NULL NUMBER(6)
 BONUS                                              NUMBER

SQL> Ayushi Tiwari
```



```
SQL> Insert into bonus(f_id)select distinct(f_id)from faculty where f_id IN (select distinct f_super from faculty);

1 row created.

SQL> select * from bonus;

      F_ID      BONUS
---------- ----------
         4       1000

SQL> Ayushi Tiwari
```

9. Add two new records to the Faculty table using the command below. These records represent new faculty who came to the university this year.

INSERT INTO faculty VALUES (5, 'Sealy', 'James', 'L', 13, '4079817153', 'Associate', 1);
INSERT INTO faculty VALUES (6, 'Smith', 'John', 'D', 10, '4238102345', 'Full', NULL);
Check the result using the *select * from faculty;* command.



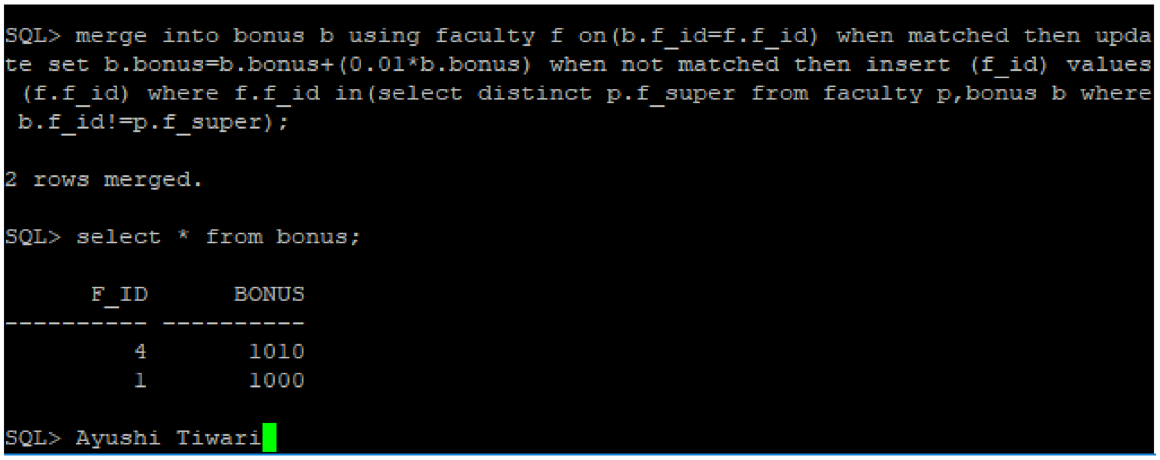
```
SQL> INSERT INTO faculty VALUES (5, 'Sealy', 'James', 'L', 13, '4079817153', 'Associate', 1);

1 row created.

SQL> INSERT INTO faculty VALUES (6, 'Smith', 'John', 'D', 10, '4238102345', 'Full', NULL);

1 row created.

SQL> select * from faculty;

      F_ID F_LAST                         F_FIRST                        F     LOC_ID F_PHONE    F_RANK       F_SUPER
---------- ------------------------------ ------------------------------ - ---------- ---------- --------- ----------
         6 Smith                          John                           D         10 4238102345 Full
         1 Marx                           Teresa                         J          9 4075921695 Associate          4
         2 Zhulin                         Mark                           M         10 4073875682 Full
         3 Langley                        Colin                          A         12 4075928719 Assistant          4
         4 Brown                          Jonnel                         D         11 4078101155 Full
         5 Sealy                          James                          L         13 4079817153 Associate          1

6 rows selected.

SQL> Ayushi Tiwari
```

10. Assume that the same Bonus table is used next year to assign and update bonuses. Use the MERGE statement to modify the Bonus table as follows:

- if a mentor already exists in the Bonus table, increase the bonus by 1%

- If there is a new mentor in the Faculty table, add him/her to the BONUS table

Check the result using the *select * from bonus;* command.



```
SQL> merge into bonus b using faculty f on(b.f_id=f.f_id) when matched then upda
te set b.bonus=b.bonus+(0.01*b.bonus) when not matched then insert (f_id) values
 (f.f_id) where f.f_id in(select distinct p.f_super from faculty p,bonus b where
 b.f_id!=p.f_super);

2 rows merged.

SQL> select * from bonus;

      F_ID      BONUS
---------- ----------
         4       1010
         1       1000

SQL> Ayushi Tiwari
```