Project Report

Databse Management System

Instructor : Prof. PM Jat

Group−7

Kushal Jangid (201351022)

Raghuvar Prajapati (201351003)

Sameer Bhati (201352013)

Rahul Nalawade (201351017)

April 21, 2015

Introduction :

Hotel Management System is a Database Management project for the course CS205 DBMS. Group #07 has successfully completed the structured design of the HMS considering the efficiency, compactibility, reliability and convenient multi-user accessibility to the system.

HMS is not only able to have a massive data storage, but also keeps the data secure and efficiently updates the database. It has been designed according to the requirements of a normal Hotel, providing general guest-services in ambience.

HMS is developed in such a manner that, it can resolve every data keeping related to a customer (guest). It can store, manipulate, remove and update data in customer's perspective.

Project Outline: Hotel Management System considers a following relations to manage data of customers from the moment he/she enters, till he/she exists.

#Relation (<attribute_list>)

- Guest (guest_id, entry_time)
- Family (guest_id, family_head_ssn, address, mobile_no, no_of_adults, no_of_children)
- Family_Member (guest_id, name, age)
- Company (guest_id, cname, location)
- Company_Member (guest_id, name, age, designation)
- Alloted (guest_id, room_no, check_in_date, check_out_date)
- Room (room_no, type, rate, status_occupied)
- Orders (guest_id, product_id, date_of_orders, time)
- Food (product_id, type, rate, name)
- Uses (guest_id, facility_id, quantity_used)
- Facility (facility_id, rate, no_of_hours, facility_type)
- Bill (guest_id, amount, payment_date, paying_method)

Hotel Management System:

- This is a real world scenario that we often see in the present world. We have created a database that will take care of the different guest with their information.

Relational Schema of the hotel management system:

Relational Scheme Diagram :

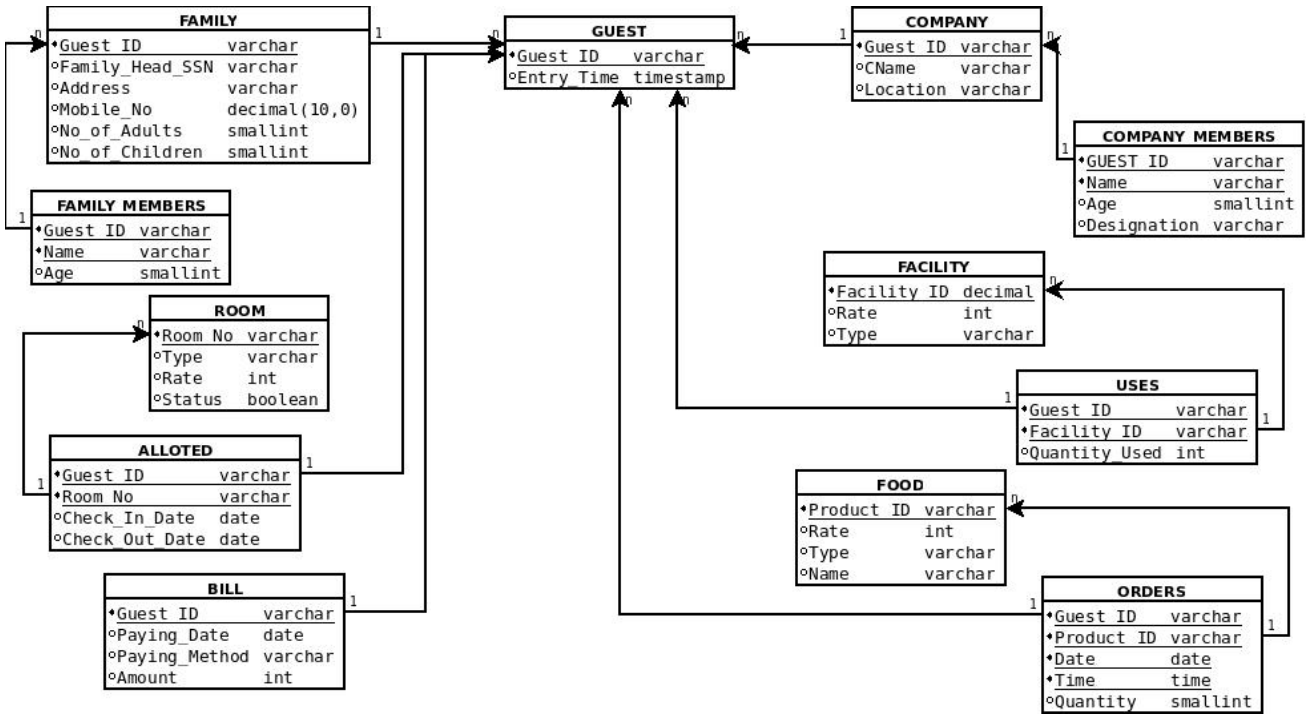

Figure 1: **Hotel Management System - Relational Schema Diagram**

DDL Script :

This is the all the tables that we have created for the database with all constraint that an attribute is follow to maintain consistency.

```
DROP SCHEMA hotel CASCADE;
CREATE SCHEMA hotel;

SET SEARCH_PATH TO hotel;

create table Family(
      Guest_ID           varchar(5) ,
      Family_Head_SSN    varchar(9) ,
      Address            varchar(50),
      Mobile_No          decimal(10,0),
      No_Of_Adults       smallint,
      No_Of_Children     smallint,
      primary key(Guest_ID),
      foreign key(Guest_ID) references Guest(Guest_ID)
       ON DELETE CASCADE
);
create table Company(
      Guest_ID            varchar(5) primary key,
      CName               varchar(40),
      Location            varchar(50),
      foreign key(Guest_ID) references Guest(Guest_ID)
         ON DELETE CASCADE
);

create table Guest(
       Guest_ID      varchar(5) primary key,
       Entry_Time    timestamp

);

create table Family_Members(
       Guest_ID          varchar(5),
       Name              varchar(40),
       Age               smallint,
       primary key(Guest_ID,Name),
       foreign key(Guest_ID) references Family(Guest_ID)
              ON DELETE CASCADE
);

create table Company_Members(
       Guest_ID    varchar(5),
       Name        varchar(40),
       Age         smallint,
       Designation varchar(40),
       primary key(Name, Guest_ID),
       foreign key(Guest_ID) references Company(Guest_ID)
               ON DELETE CASCADE
);

create table Room(
```

```
      Room_No        varchar(5) primary key,
      Type           varchar(50),
      Rate           int,
      Status_Occupied boolean
);

create table Alloted(
      Guest_ID       varchar(5),
      Room_No        varchar(5),
      Check_in_date  Date,
      Check_out_date Date,
      primary key(Guest_ID, Room_No),
      foreign key(Guest_ID) references Guest(Guest_ID)
              ON DELETE CASCADE,
      foreign key(Room_No) references Room(Room_No)
              ON DELETE CASCADE
);

create table BILL (
      Guest_ID         varchar(5) primary key,
      Amount           int,
      Payment_Date     Date,
      Paying_Method    varchar(20),
      foreign key(Guest_ID) references Guest(Guest_ID)
              ON DELETE CASCADE
);

CREATE TABLE Facility (
   Facility_id varchar(5) primary key,
   Rate    int,
   No_of_hours    int,
   Facility_type varchar(40)

);

create table uses (
       Guest_ID      varchar(5),
       Facility_id   varchar(5),
       Quantity_used int,
       primary key(Guest_ID, Facility_id),
       foreign key(Guest_ID) references Guest(Guest_ID)
              ON DELETE CASCADE,
       foreign key(Facility_id) references Facility(Facility_id)
              ON DELETE CASCADE
);


create table Food (
       Product_ID  varchar(5) primary key,
       Rate        int,
       Type        varchar(60),
       Name        varchar(60)
);
```

```
create table Orders (
        Guest_ID        varchar(5),
        Product_ID      varchar(5),
        Date_Of_Orders Date,
        Time            TIME,
        Quantity        smallint,
        primary key(Guest_ID, Product_ID, Date_of_Orders, Time),
        foreign key(Guest_ID) references Guest(Guest_ID)
                ON DELETE CASCADE,
        foreign key(Product_ID) references Food(Product_ID)
                ON DELETE CASCADE
);
```

ER Diagram :

This is the ER Diagram for database that made relation more understandable.

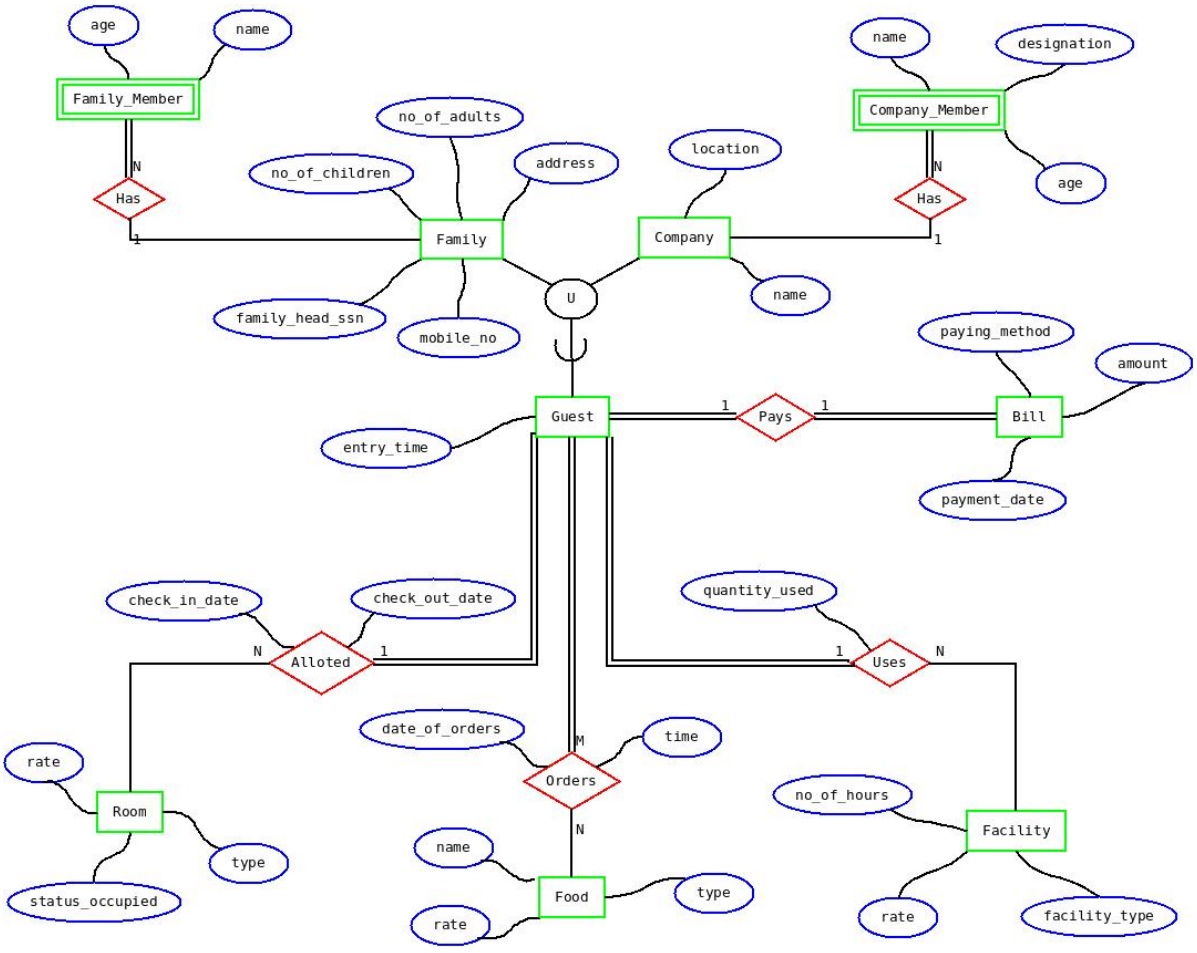

Figure 2: **Hotel Management System - Relational Schema Diagram**

Relations and their Functional Dependencies

Functional Dependencies for all the relations:

1- **Guest**(guest_id , entity_time)
FD- guest_id → { entity_time }

2- **Company**(guest_id, cname, location)
FD- guest_id → { cname }
guest_id → { location }

3- **Company_Members**(guest_id, name, age, designation)
FD- { guest_id , name }→ { age }
{ guest_id, name }→ { designation }

4- **Family**(guest_id , head_ssn , address, mobile_no. , adults , no_of_children)
FD- guest_id → { head_ssn }
guest_id → { mobile_no. }
guest_id → { adults }
guest_id → { no_of_children }

5- **Family_Members** (guest_id , name , age)
FD- { guest_id , name } → { age }

6- **Alloted**(guest_id , room_id , check_in_date , check_out_date)
FD- { guest_id , room_id } → { check_in_date , check_out_date }

7- **Room** (room_no , type, rate, status_occupied)
FD- room_no → { type, rate, status_occupied }

8- **Facility** (facility_id , no_of_hours , rate , type)
FD- facility_id → { no_of_hours , rate , type }

9- **Food**(product_id , rate , type , name)
FD- product_id → { rate , type , name }

10- **Uses**(guest_id, facility_id , quantity_used)
FD- { guest_id , facility_id } → { quantity_used }

11- **Orders**(guest_id , product_id , date_of_time , time , quantity)
FD- { guest_id , product_id , date_of_time } → { quantity }

12- **Bill**(guest_id , amount, paying_method, paying_date)
FD- guest_id → { amount, paying_method, paying_date }

Relational Algebra AND SQL Queries:

Here we are giving the queries that our database is supposed to answer:

Qu-1 List out the total numbers of vacant rooms ?

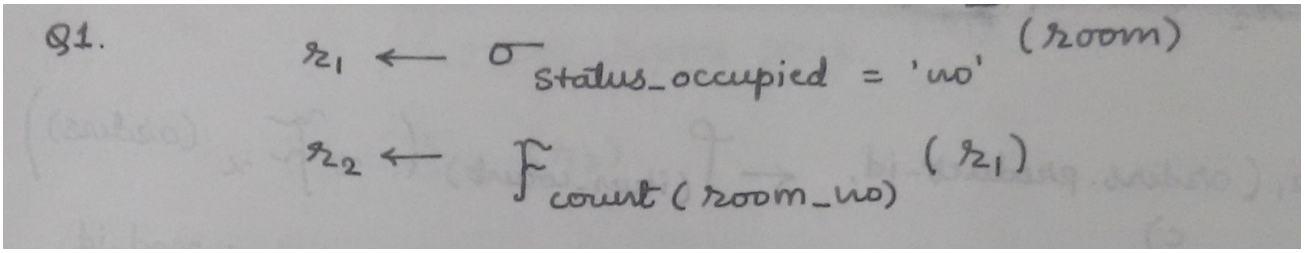

Figure 3: **Relational Algebra Expression**

Qu-2 List out the total numbers of filled rooms ?

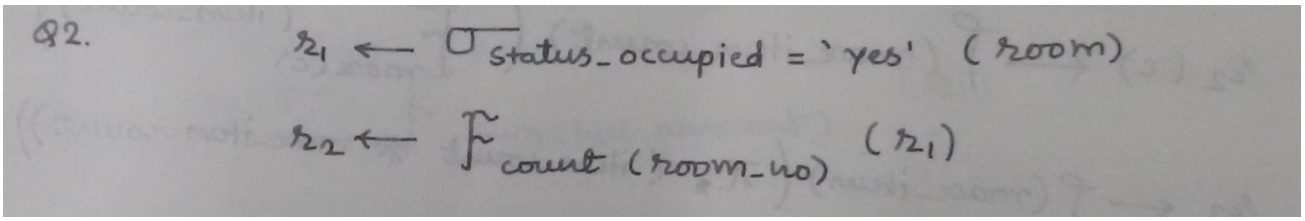

Figure 4: **Relational Algebra Expression**

Qu-3 List the guest id with their room no, tpye of the room ?

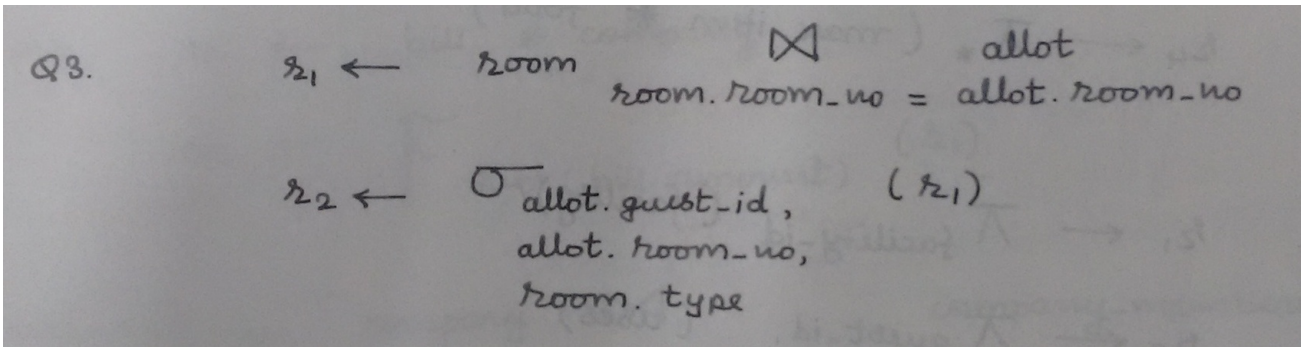

Figure 5: **Relational Algebra Expression**

Qu-4 List the guest id who only had food in hotel ?

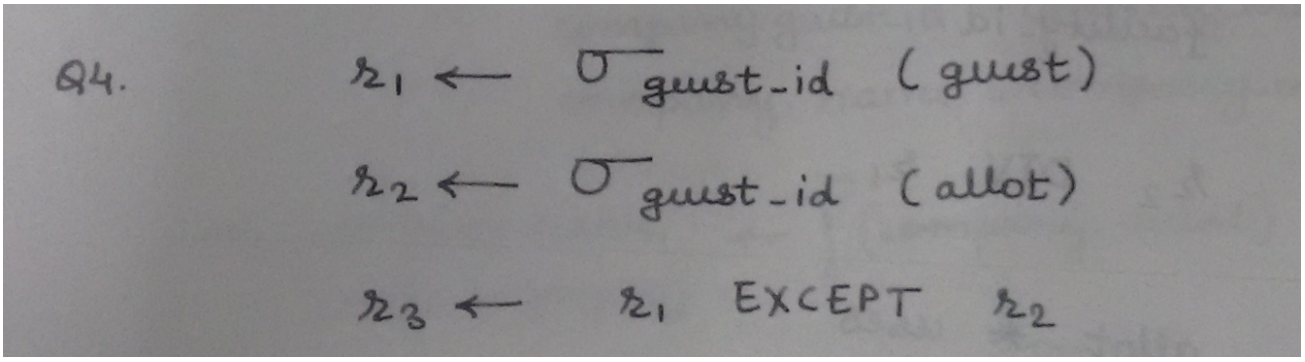

Figure 6: **Relational Algebra Expression**

Qu-5 Find out the details of the very first customer of the hotel ?

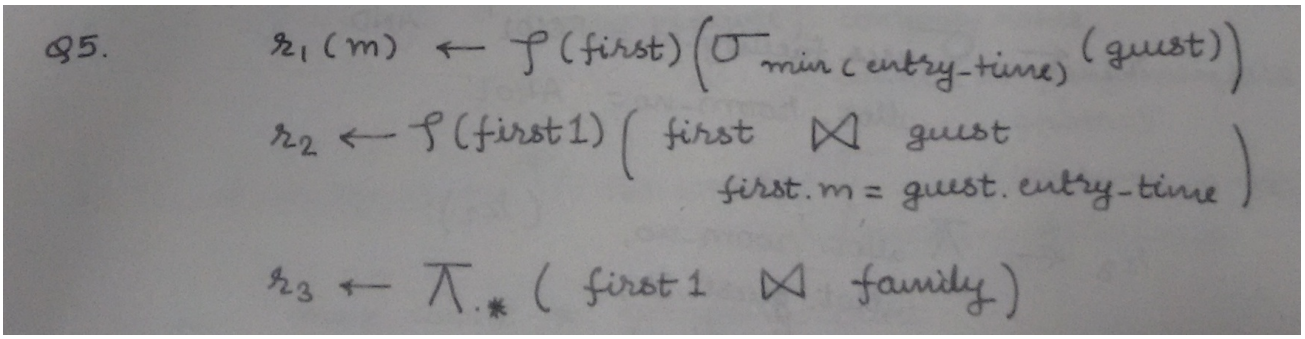

Figure 7: **Relational Algebra Expression**

Qu-6 List out the number of items in each type of food ?

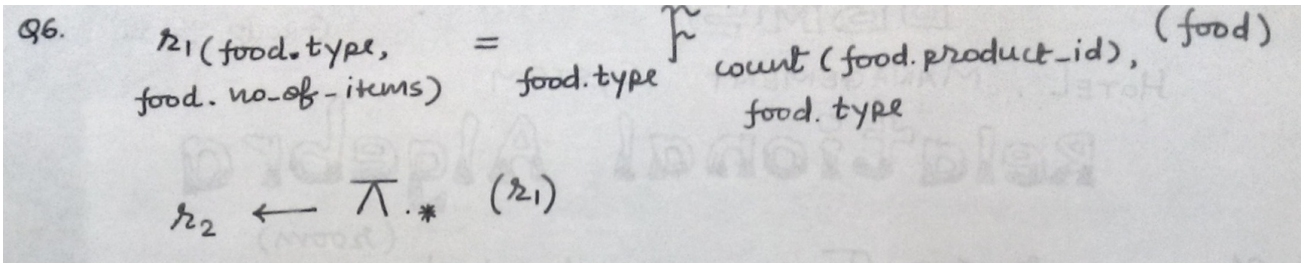

Figure 8: **Relational Algebra Expression**

Qu-7 List out the most ordered food items ate by the customers ?

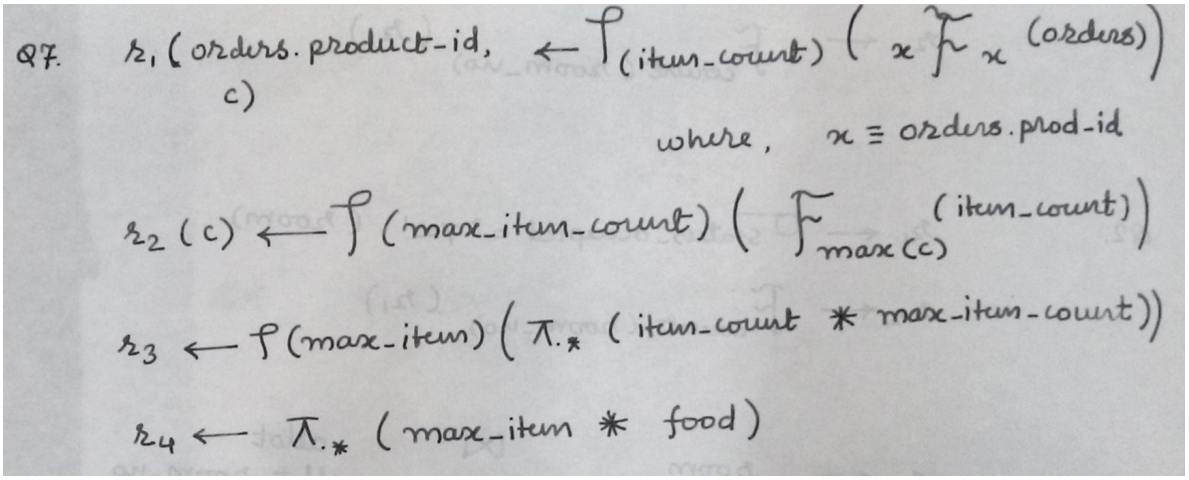

Figure 9: **Relational Algebra Expression**

Qu-8 List out guest id which uses all types of facilities ?

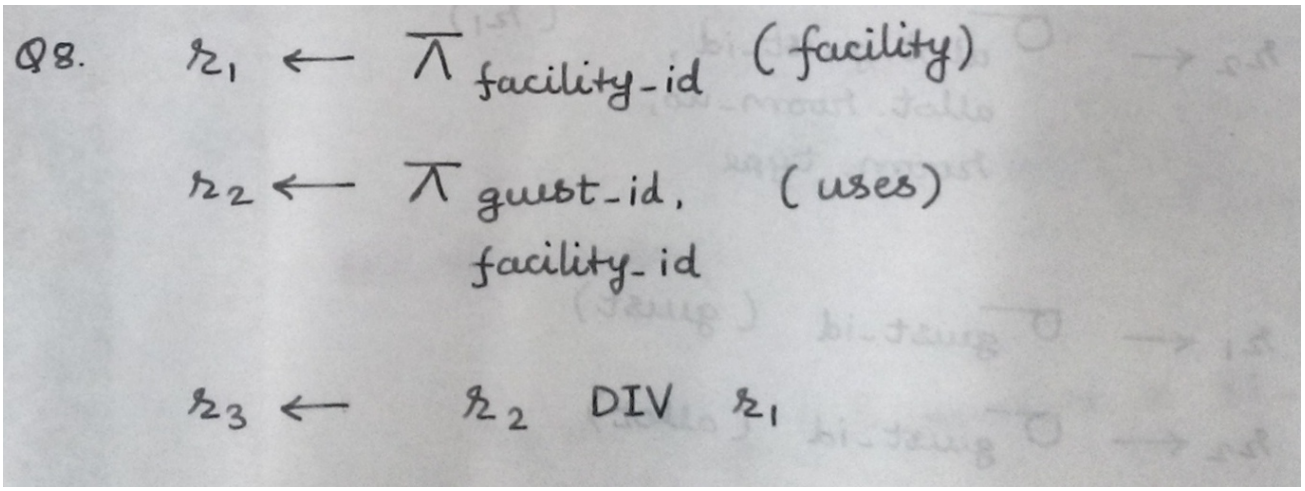

Figure 10: **Relational Algebra Expression**

Qu-10 List out the guest id who were alloted more than or equal to 2 rooms ?

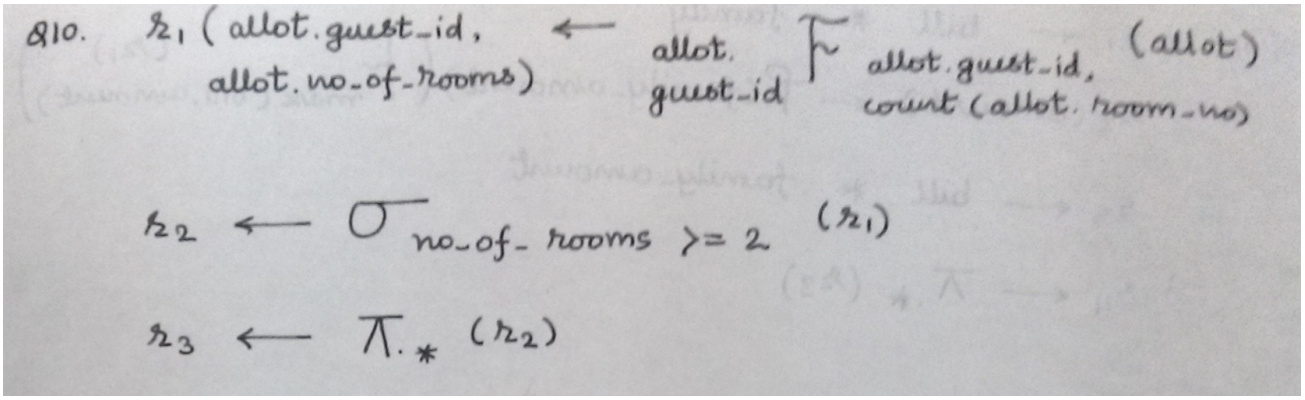

Figure 11: **Relational Algebra Expression**

Qu-11 Total Amount paid by the family type customers ?

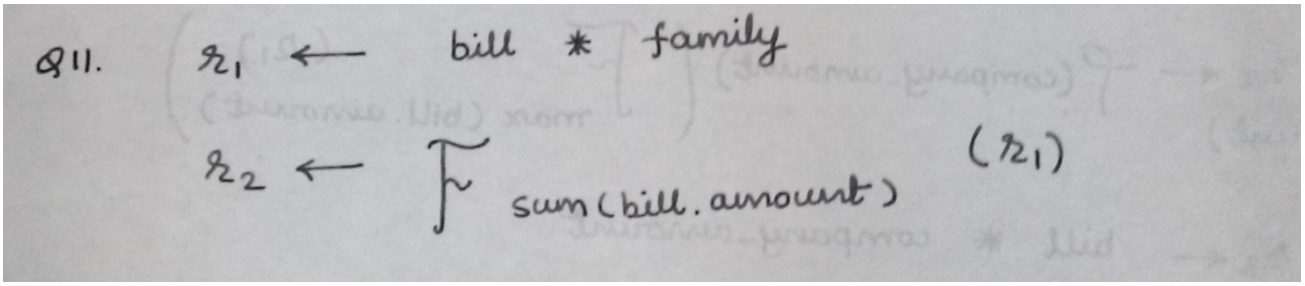

Figure 12: **Relational Algebra Expression**

Qu-12 Total Amount paid by the company type customers ?

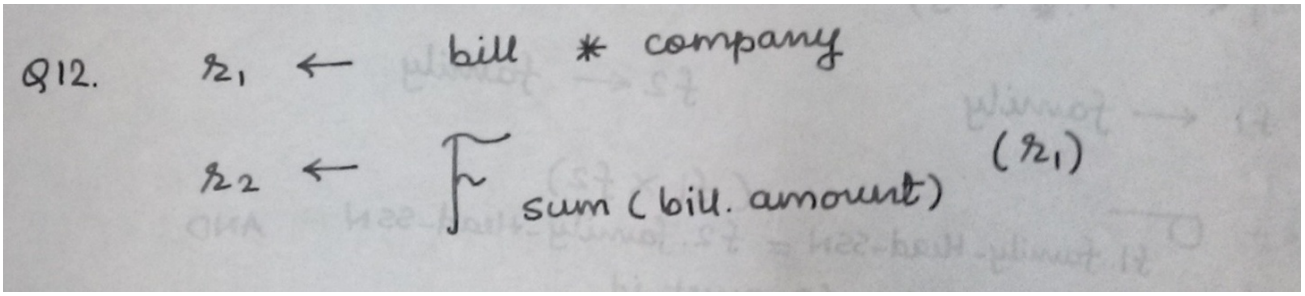

Figure 13: **Relational Algebra Expression**

Qu-13 Find out the name of that company that came with maximum number of employee ?

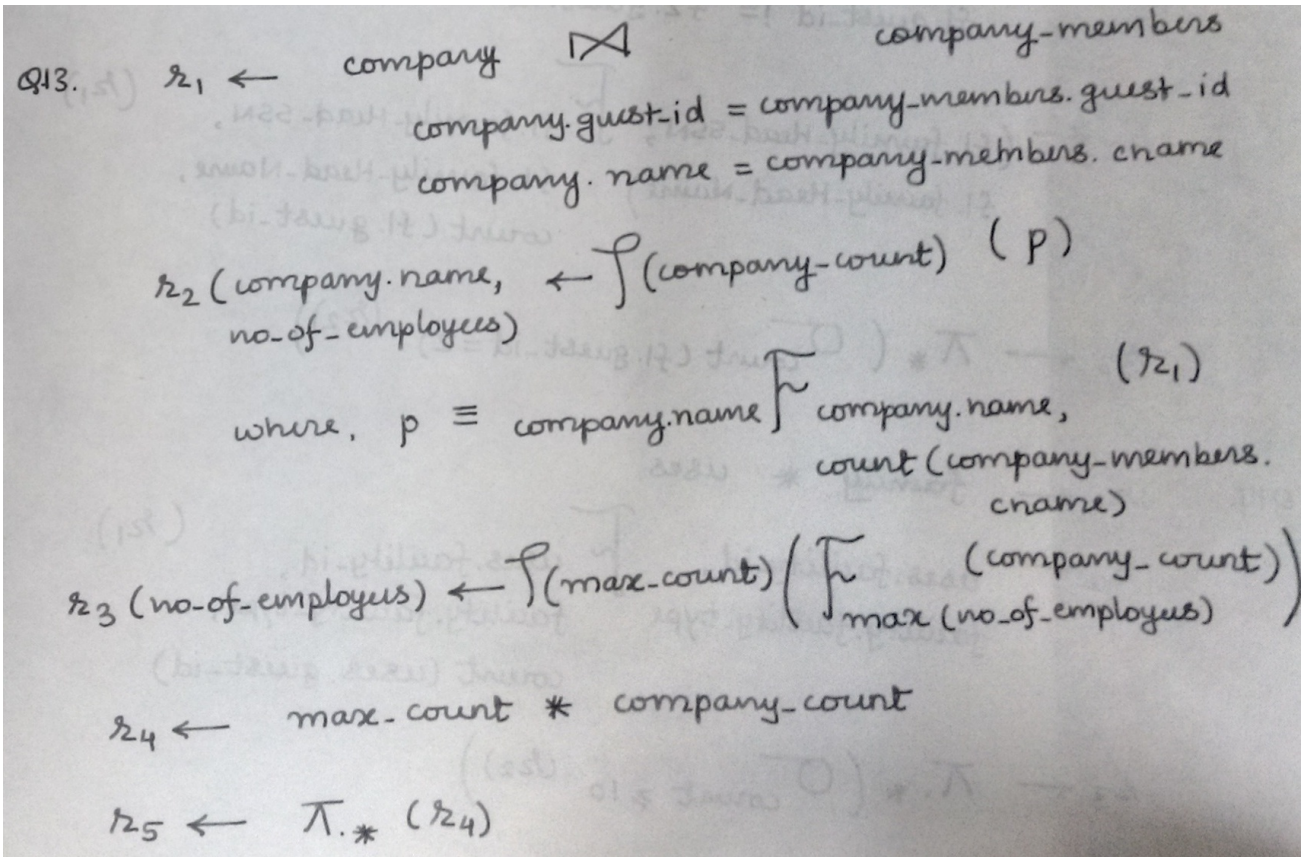

Figure 14: **Relational Algebra Expression**

Qu-14 Most Profitable guest id that is of family type customer ?

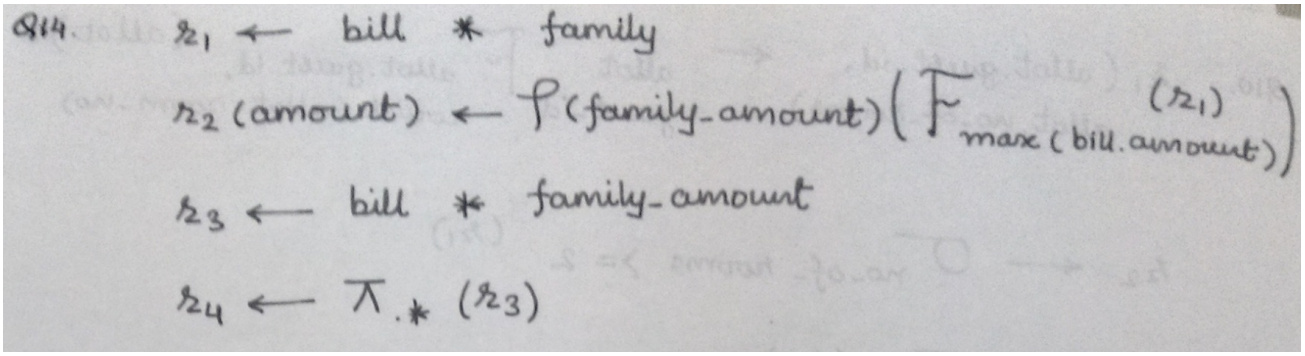

Figure 15: **Relational Algebra Expression**

Qu-15 Most Profitable guest id that is of company type customer ?

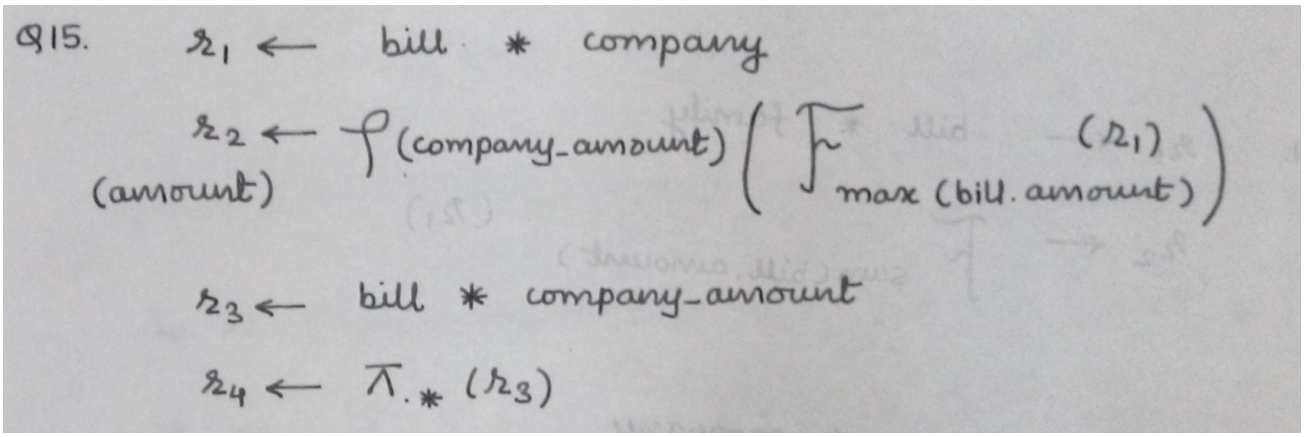

Figure 16: **Relational Algebra Expression**

Qu-16 List out the Head SSN with Head Name who came twice to the hotel ?

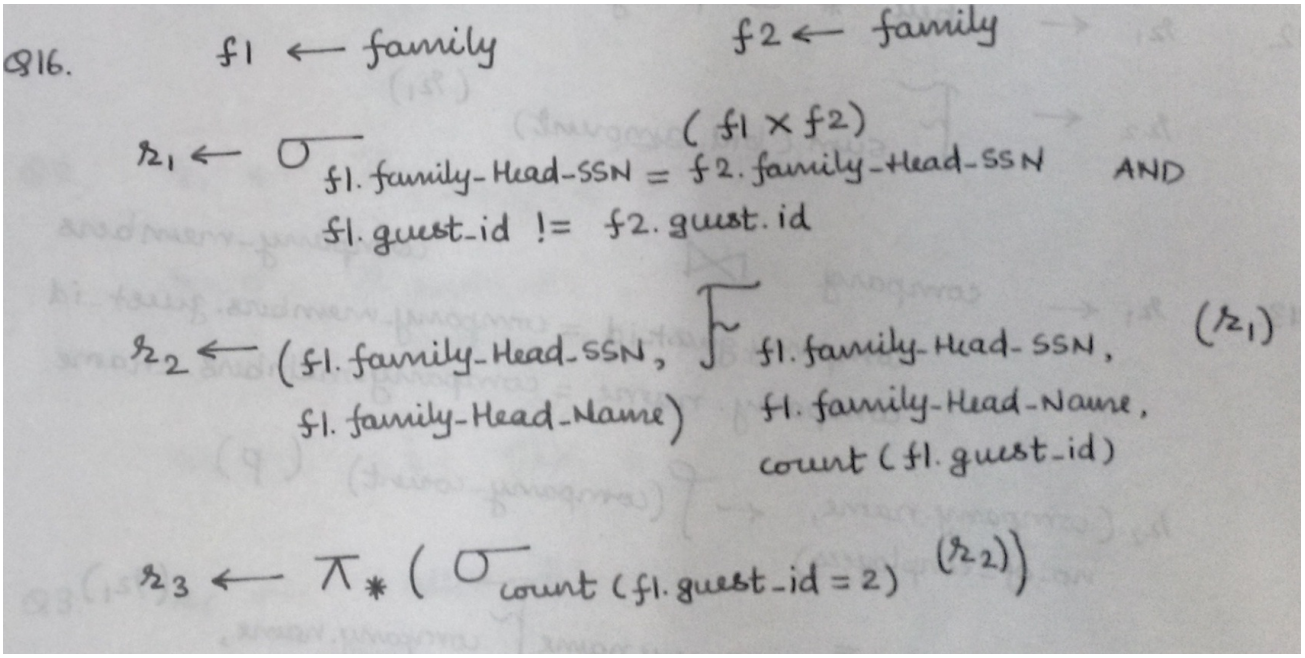

Figure 17: **Relational Algebra Expression**

Qu-17 List the falility used by more than 10 customers ?

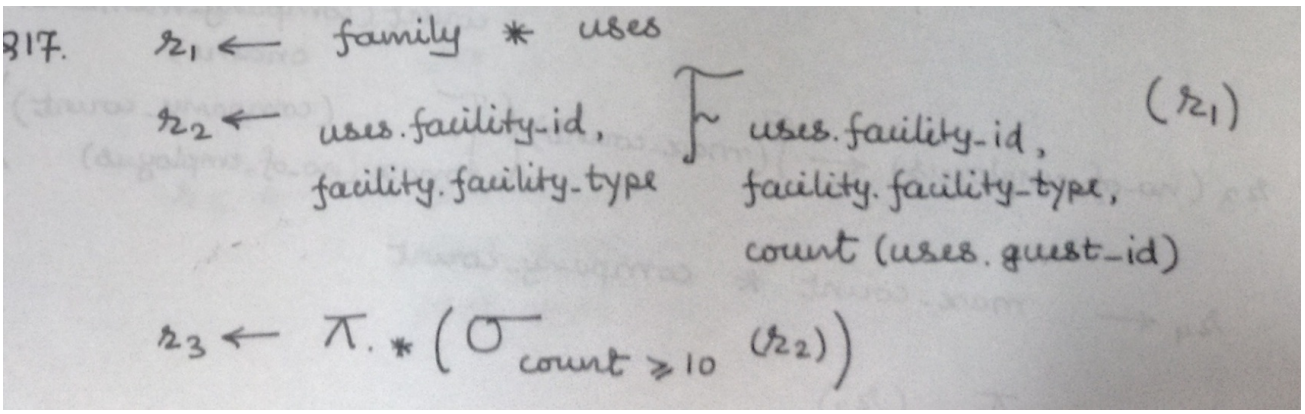

Figure 18: **Relational Algebra Expression**

Qu-18 List out the different company names with their number of employees ?

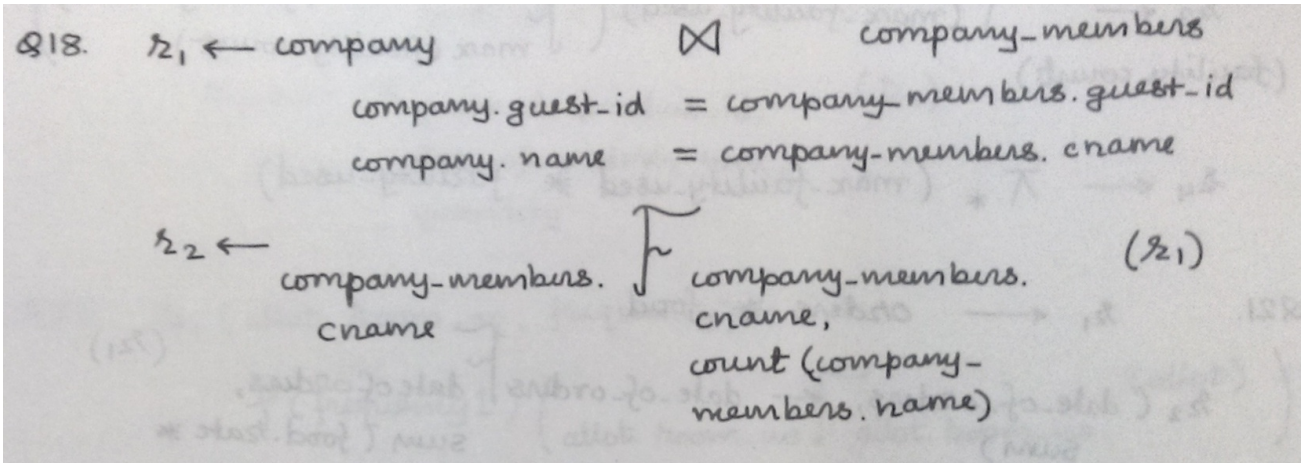

Figure 19: **Relational Algebra Expression**

Qu-19 List out the guest id of the family type with the maximum number of members with them ?

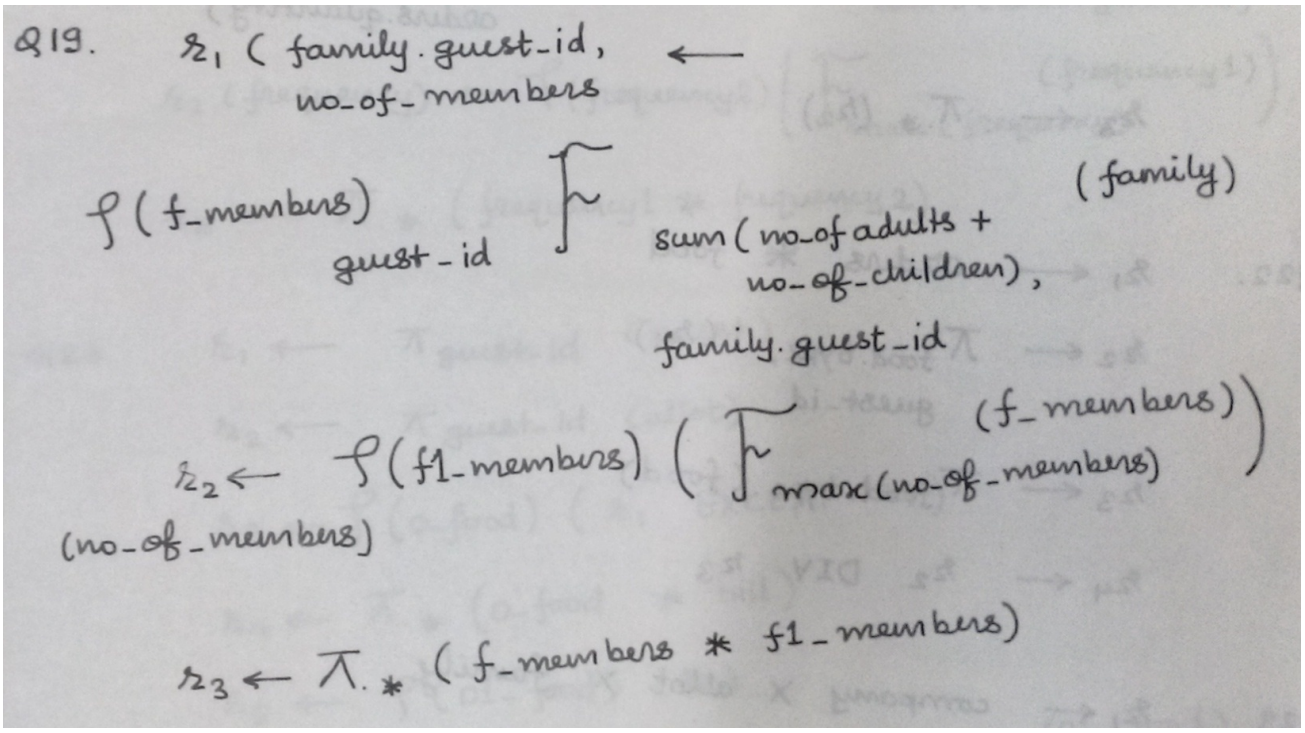

Figure 20: **Relational Algebra Expression**

Qu-20 List out the facility used by the most number of customers ?

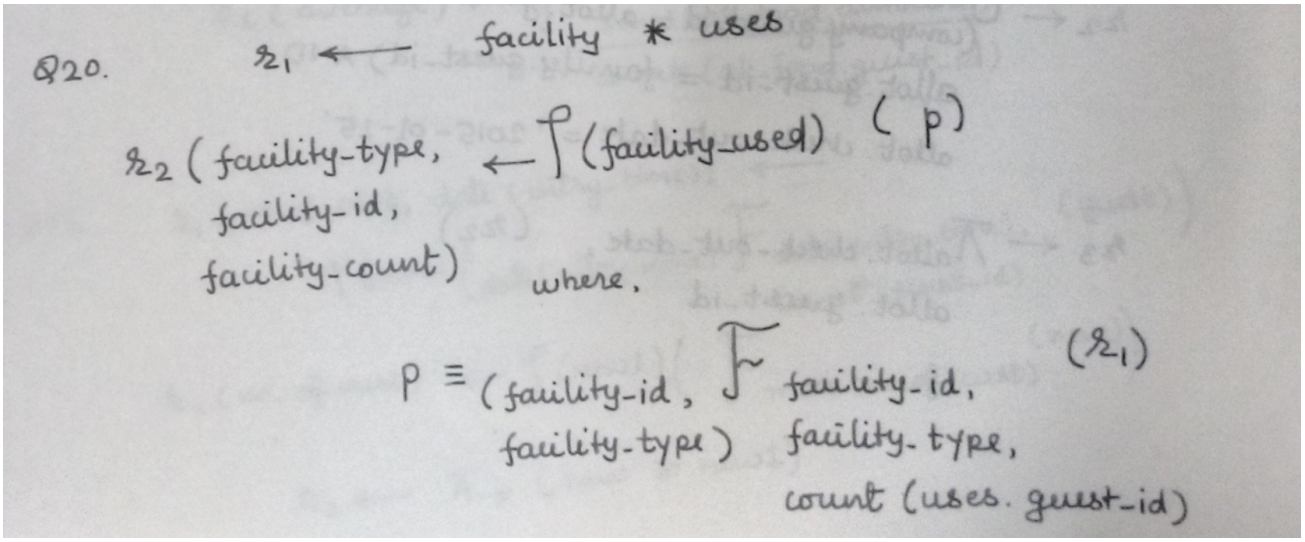

Figure 21:

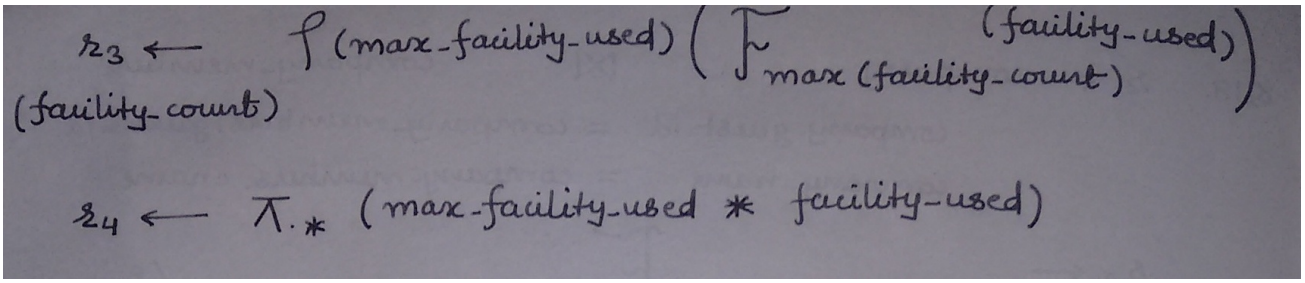

Figure 22: **Relational Algebra Expression**

Qu-21 Find out the total food amount on a particular date given by the customers ?

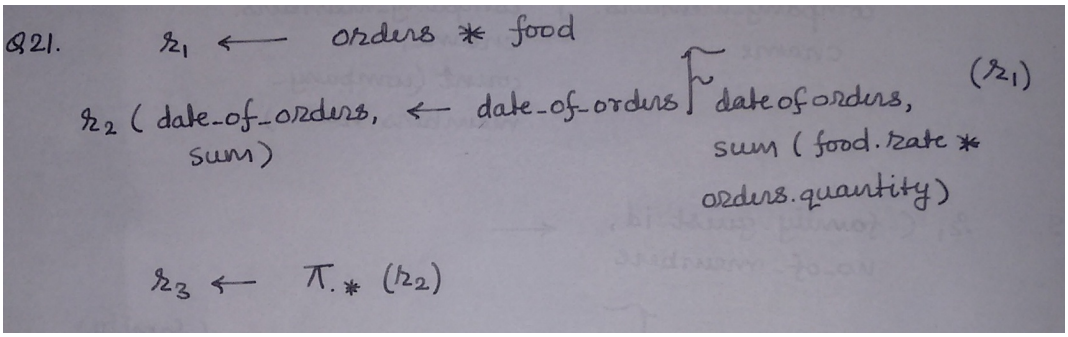

Figure 23: **Relational Algebra Expression**

Qu-22 List out the guest id that orders all types of foods ?

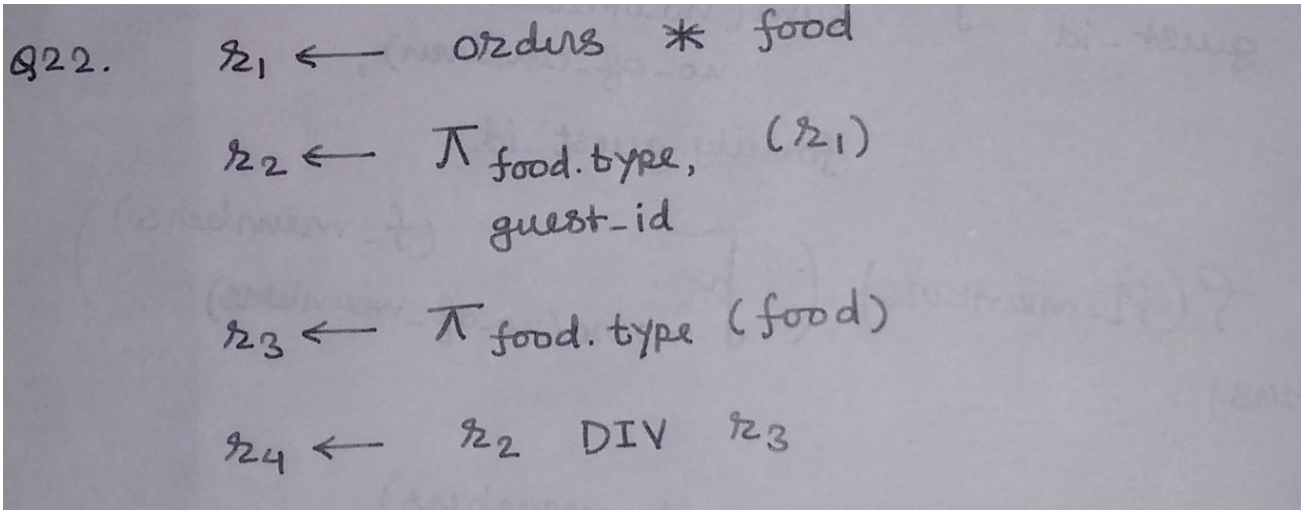

Figure 24: **Relational Algebra Expression**

Qu-23 List out the total number of people who checked out on 11-01-2015 ?

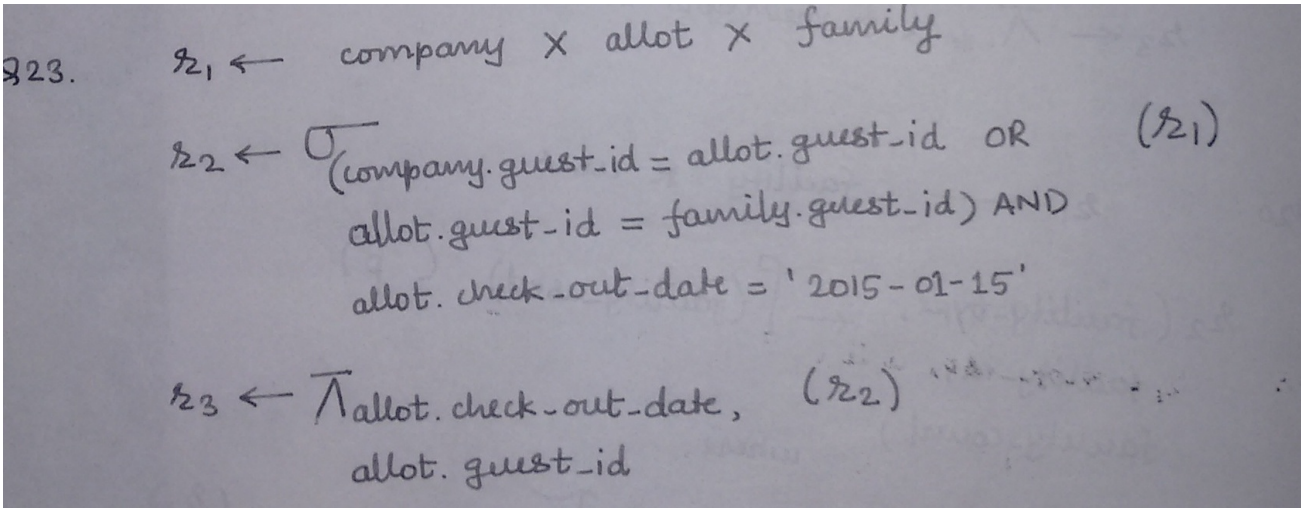

Figure 25: **Relational Algebra Expression**

Qu-24 List out the orders for the guest id - C1004 during his period of living ?

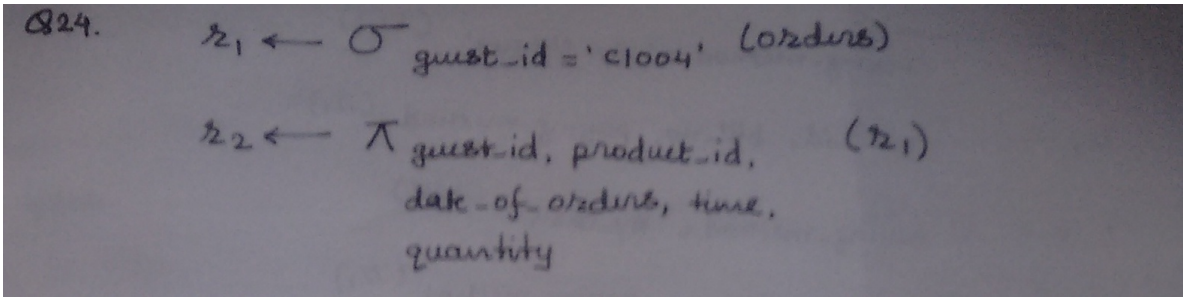

Figure 26: **Relational Algebra Expression**

Qu-25 List out the most frequently alloted room ?

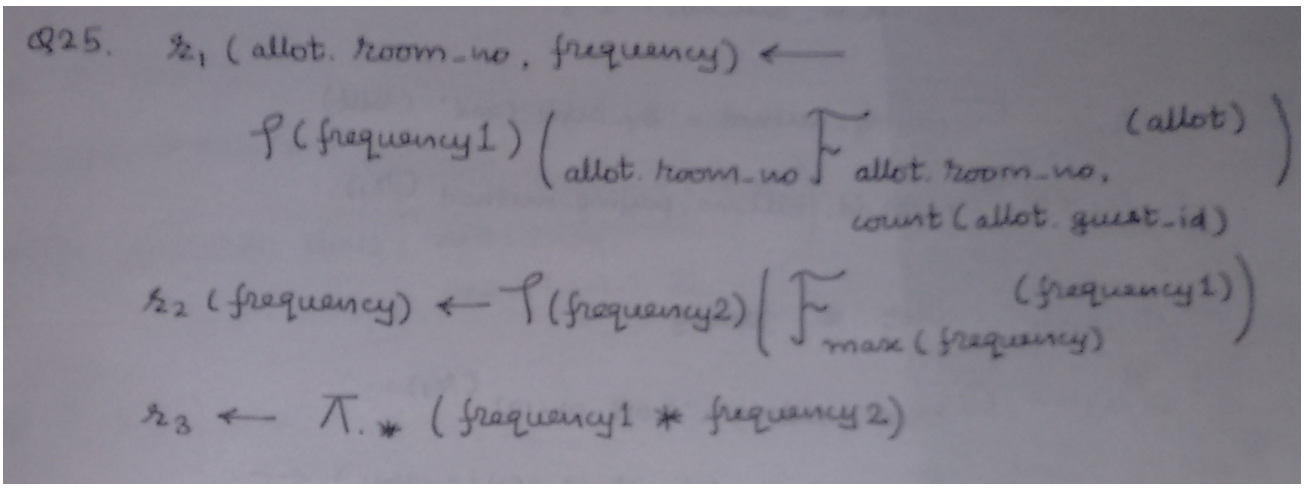

Figure 27: **Relational Algebra Expression**

Qu-26 Find out the average amount of bill paid by guests visting only for food ?

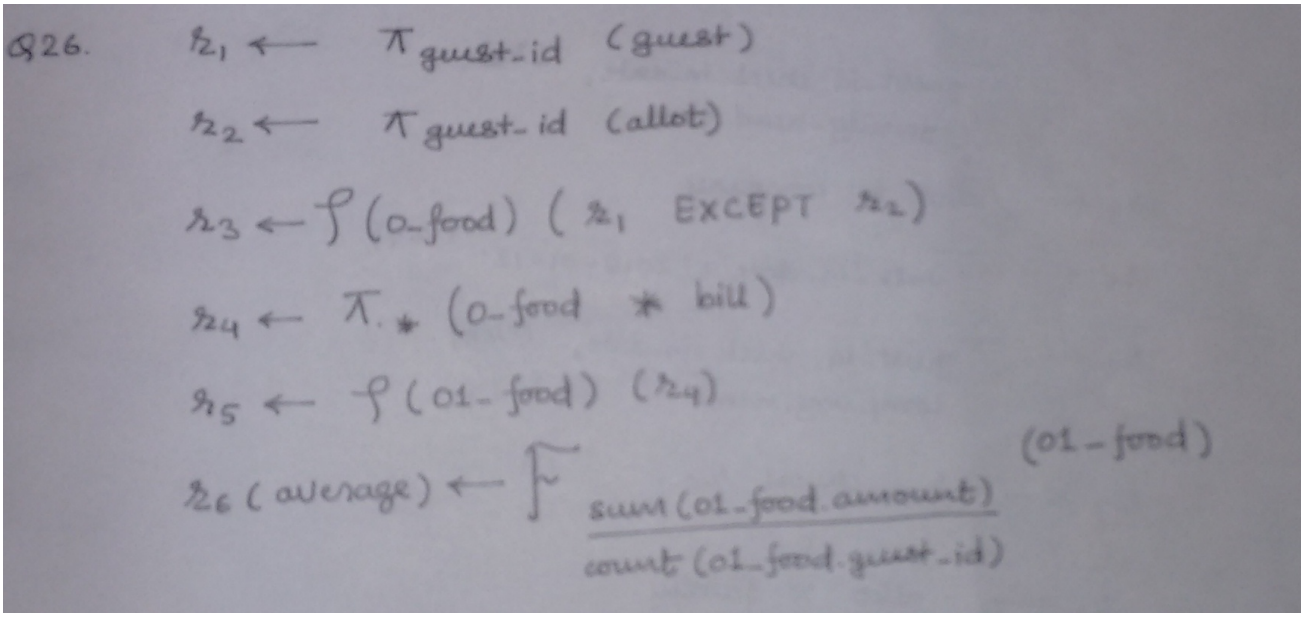

Figure 28: **Relational Algebra Expression**

Qu-27 Find out the date on which the maximum number customers came to hotel ?

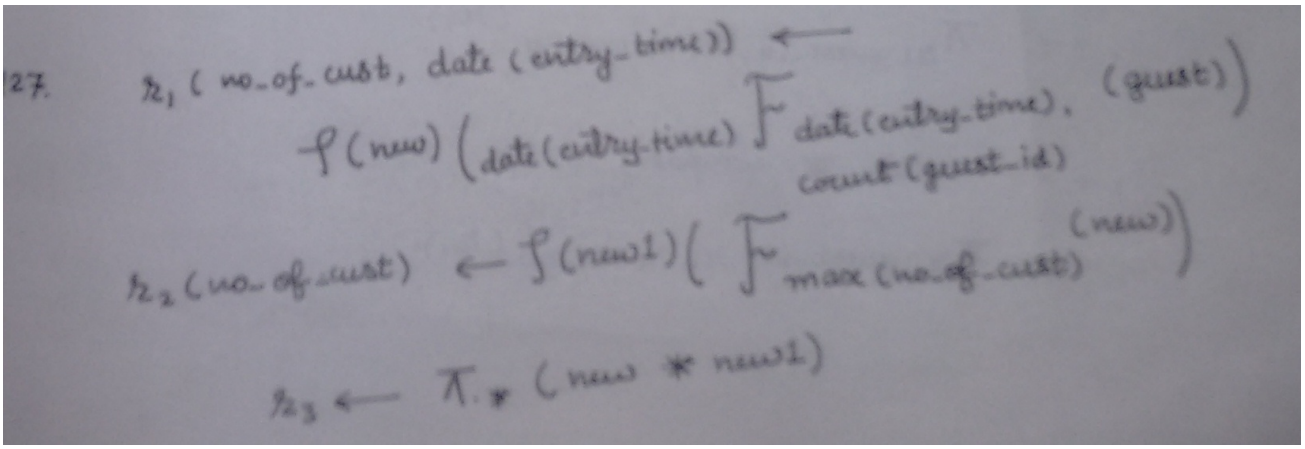

Figure 29: **Relational Algebra Expression**

Qu-28 List out the guest id that paid there bills by Cheque?

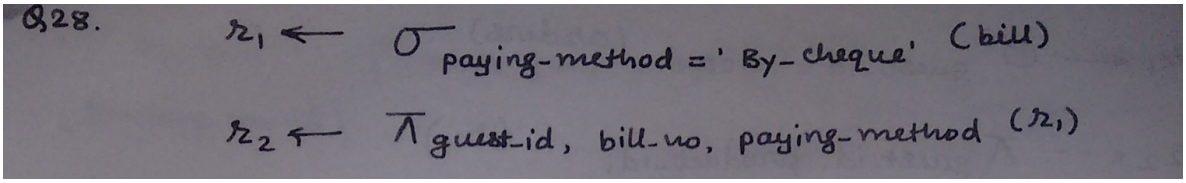

Figure 30: **Relational Algebra Expression**

Qu-29 List out the guest id that paid there bills by Cash?

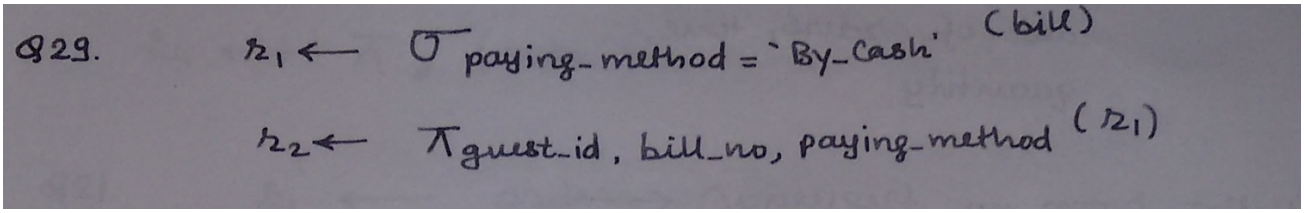

Figure 31: **Relational Algebra Expression**

Qu-30 List out the guest id that paid there bills by Debit-Card ?

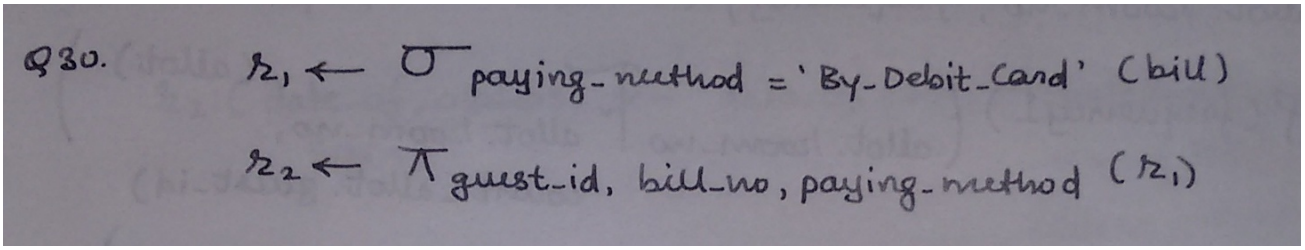

Figure 32: **Relational Algebra Expression**

Qu-31 List name, guest id of families and company check in on 12-01-2015 ?

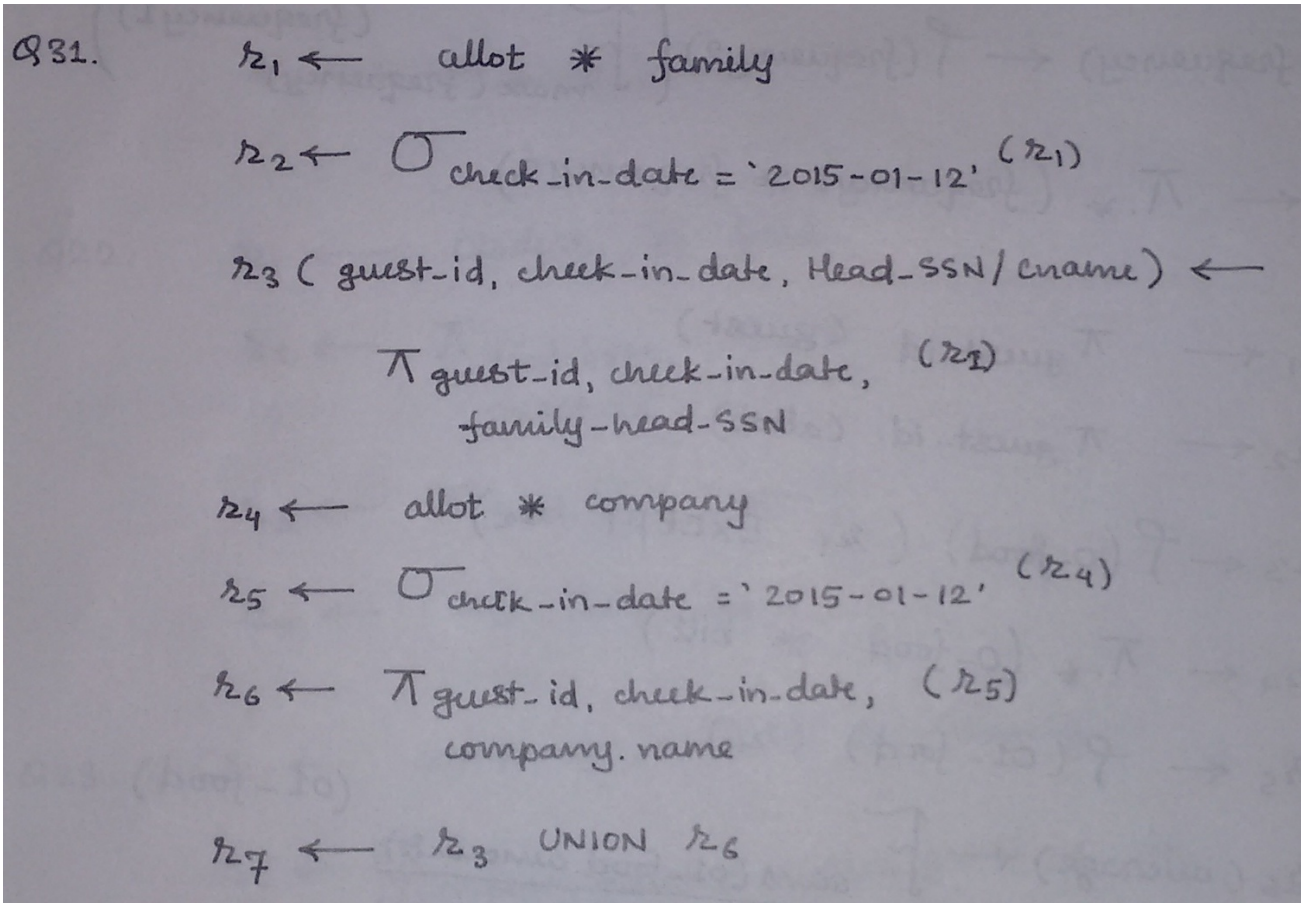

Figure 33: **Relational Algebra Expression**

Qu-32 List out guest id with their room-no. that are of family type ?

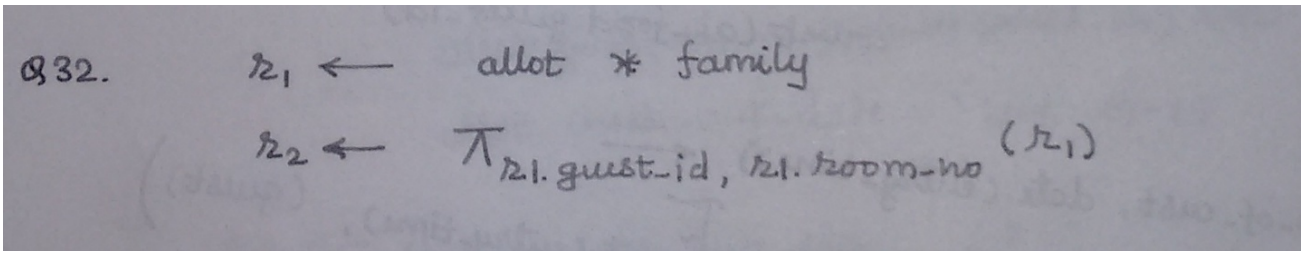

Figure 34: **Relational Algebra Expression**

Qu-33 List out guest id with their room-no. that are of company type ?

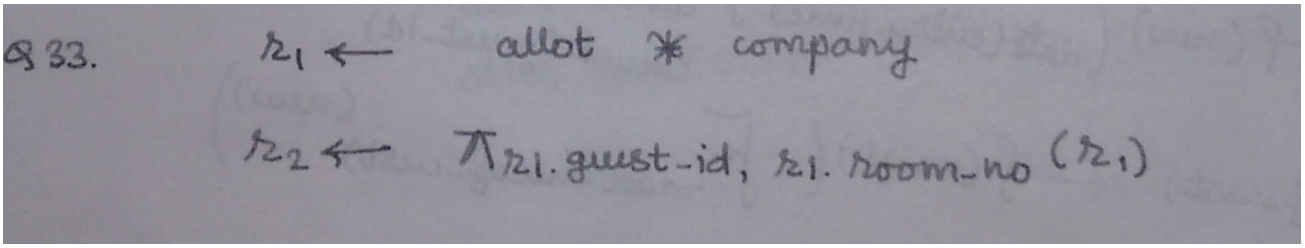

Figure 35: **Relational Algebra Expression**

Qu-34 List out the facility id used by guest that lived in room-no = A101 ?

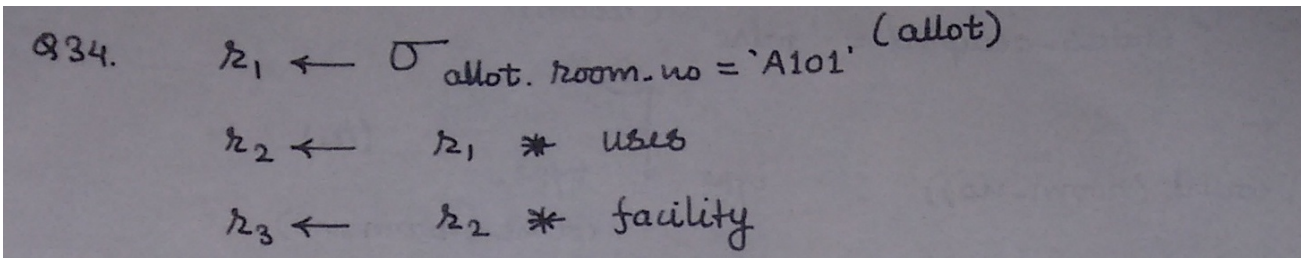

Figure 36: **Relational Algebra Expression**

Qu-35 List out the date on which maximum number of customer of family tpye came to hotel ?

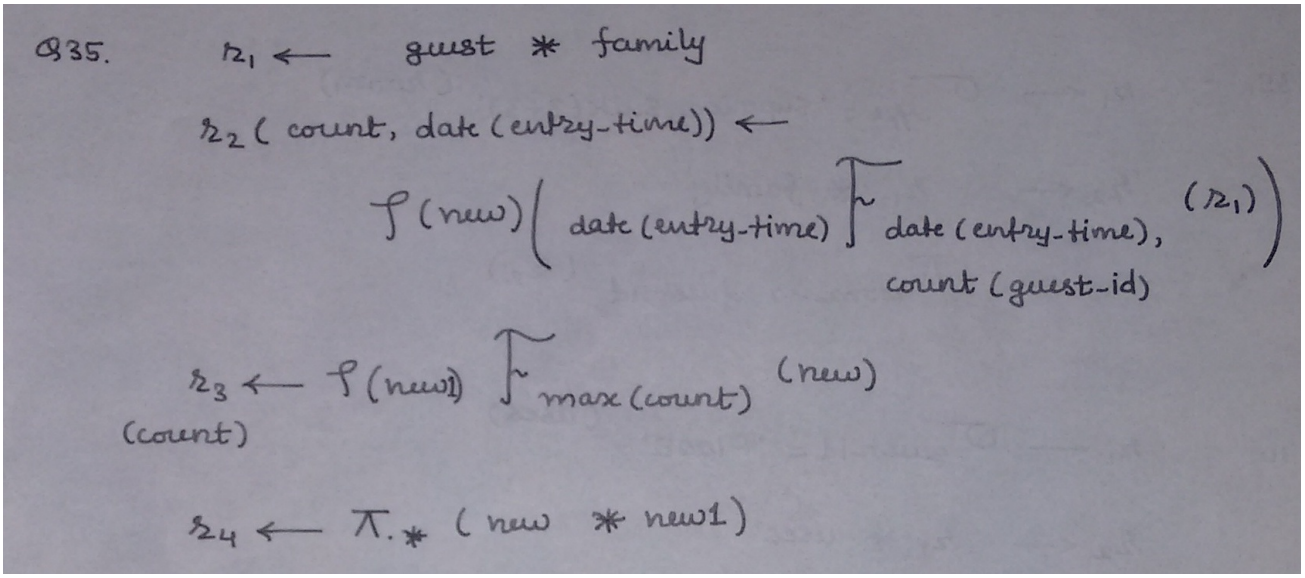

Figure 37: **Relational Algebra Expression**

Qu-36 List out the date on which maximum number of customer of company tpye came to hotel ?

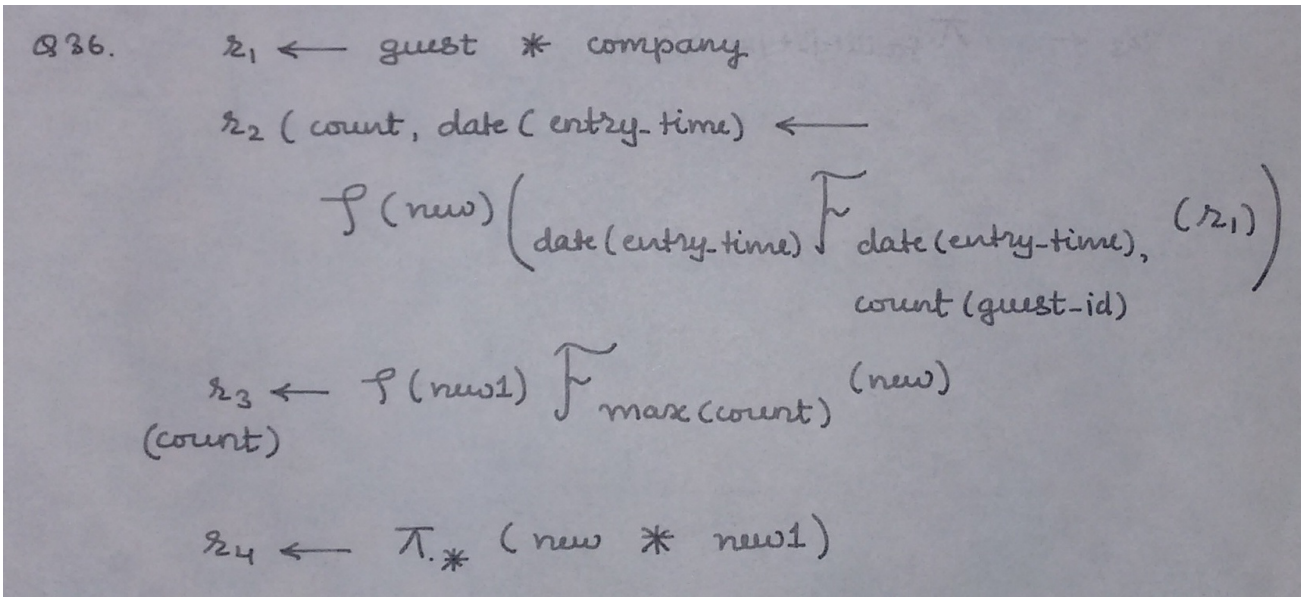

Figure 38: **Relational Algebra Expression**

SQL Queries:

Qu_1- List out the total numbers of vacant rooms ?

Ans- SQl-

```
SELECT      count(room.room_no)
FROM        hotel.room
WHERE    room.status_occupied = no;
```

Output-

	count bigint
1	52

Figure 39: **Executed output table**

Qu_2- List out the total numbers of filled rooms?

Ans- SQL-

```
SELECT     count(room.room_no)
FROM       hotel.room
WHERE   room.status_occupied = yes;
```

Output-

	count bigint
1	5

Figure 40: **Executed output table**

Qu_3- List the guest_id with their room_no, tpye of the room?

Ans- SQL-

```
SELECT  alloted.guest_id,
        alloted.room_no,
        room.type
  FROM  hotel.room,
        hotel.alloted
 WHERE  alloted.room_no = room.room_no;
```

Output-

	guest_id character varying(5)	room_no character varying(5)	type character varying(50)
1	C1001	A401	Executive Suite(5+0
2	C1001	A402	Executive Suite(5+0
3	F1004	A002	Single Suite(1+1)
4	C1002	A403	Executive Suite(5+0
5	C1002	A404	Executive Suite(5+0
6	C1003	A405	Executive Suite(5+0
7	C1003	A406	Executive Suite(5+0
8	C1004	A501	Executive Suite(5+0
9	C1004	A502	Executive Suite(5+0
10	C1005	A503	Executive Suite(5+0
11	C1005	A504	Executive Suite(5+0
12	C1006	A505	Executive Suite(5+0
13	C1006	A506	Executive Suite(5+0
14	C1007	A401	Executive Suite(5+0
15	C1007	A402	Executive Suite(5+0
16	C1008	A403	Executive Suite(5+0
17	C1008	A404	Executive Suite(5+0
18	C1009	A405	Executive Suite(5+0
19	C1010	A406	Executive Suite(5+0
20	C1011	A501	Executive Suite(5+0
21	C1012	A502	Executive Suite(5+0
22	C1012	A503	Executive Suite(5+0
23	C1013	A504	Executive Suite(5+0
24	C1014	A505	Executive Suite(5+0
25	C1015	A506	Executive Suite(5+0
26	C1016	A401	Executive Suite(5+0
27	C1017	A402	Executive Suite(5+0
28	C1017	A403	Executive Suite(5+0
29	C1018	A404	Executive Suite(5+0
30	C1019	A405	Executive Suite(5+0
31	C1020	A406	Executive Suite(5+0
32	C1021	A501	Executive Suite(5+0
33	C1021	A502	Executive Suite(5+0
34	C1022	A503	Executive Suite(5+0
35	C1023	A504	Executive Suite(5+0
36	C1024	A505	Executive Suite(5+0
37	C1025	A401	Executive Suite(5+0
38	C1025	A402	Executive Suite(5+0
39	F1025	A202	Prestige Junior Sui
40	F1021	A009	Single Suite(1+1)
41	F1019	A008	Single Suite(1+1)
42	F1017	A006	Single Suite(1+1)
43	F1016	A005	Single Suite(1+1)
44	F1014	A201	Prestige Junior Sui
45	F1012	A301	Superior Suite(3+3)
46	F1011	A104	Junior Suite(2+2)
47	F1008	A103	Junior Suite(2+2)
48	F1007	A003	Single Suite(1+1)
49	F1006	A102	Junior Suite(2+2)
50	F1005	A101	Junior Suite(2+2)

Figure 41: **Executed output table**

Qu_4- List the guest_id who only had food in hotel?

Ans- SQL-

```
   SELECT  guest.guest_id
   FROM    hotel.guest
EXCEPT
   SELECT  alloted.guest_id
   FROM    hotel.alloted;
```

Output-

	guest_id character varying(5)
1	F1010
2	F1020
3	F1003
4	F1018
5	F1001
6	F1015
7	F1013
8	F1022
9	F1002
10	F1023
11	F1009
12	F1024

Figure 42: **Executed output table**

Qu_5- Find out the details of the very first customer of the hotel?

Ans- SQL-

```
SELECT     .*
FROM ( SELECT guest.guest_id
      FROM (SELECT min(guest.entry_time) AS m
      FROM hotel.guest ) AS first
      JOIN hotel.guest ON first.m = guest.entry_time ) AS
          first1 NATURAL JOIN hotel.family;
```

Output-

	guest_id character varying(5)	family_head_ssn character varying(9)	family_head_name character varying(50)	address character varying(50)	mobile_no numeric(10,0)	no_of_adults smallint	no_of_children smallint
1	F1001	222334455	Raman Lamba	Delhi, India	9982543567	2	1

Figure 43: **Executed output table**

Qu_6- List out the number of items in each type of food?

Ans- SQL-

```
SELECT  food.type,
        count(food.product_id) AS no_of_items
FROM       hotel.food
GROUP BY   food.type
```

Output-

	type character varying(60)	no_of_items bigint
1	Sabzi	10
2	Soups & Salads	6
3	Sammundaree Namoone	5
4	Murgi	13
5	Chutneys	7
6	Paneer	9
7	Continental/Chinese	13
8	Beverages	10
9	Dosa	7
10	Desserts	6
11	Tandoor	7
12	Dal	6
13	Appetizers	10
14	Gosht	12
15	Biryani	7
16	Rice	6
17	Breads	11

Figure 44: **Executed output table**

Qu_7- List out the most ordered food items ate by the customers?

Ans- SQL-

```
DROP TABLE item_count;
CREATE TABLE item_count AS
(SELECT orders.product_id,
      count(orders.guest_id) AS c
FROM       hotel.orders
GROUP BY   orders.product_id );
DROP TABLE    max_item_count;
CREATE TABLE  max_item_count AS
(SELECT max(c) AS c FROM item_count );
DROP TABLE    max_item;
CREATE TABLE  max_item AS
( SELECT *
   FROM item_count NATURAL JOIN max_item_count );
SELECT *
FROM  hotel.food NATURAL JOIN max_item
```

Output-

	product_id character varying(5)	rate integer	type character varying(60)	name character varying(60)	c bigint
1	P0099	90	Rice	Zeera Rice	11
2	P0018	29	Breads	Tandoori Roti	11

Figure 45: **Executed output table**

Qu_8- List out guest_id which uses all types of facilities?

Ans- SQL-

```
SELECT *
FROM  hotel.guest AS g
WHERE NOT EXISTS (( SELECT f.facility_id
FROM  hotel.facility AS f )
EXCEPT ( SELECT u.facility_id
           FROM  hotel.uses AS u
           WHERE u.guest_id = g.guest_id
           ));
```

Output-

	guest_id character varying(5)	entry_time timestamp without time zone

Figure 46: **Executed output table**

Qu_9- List guest_id that lived in room_no = 'A401' and used facility_id = 'FC001?

Ans- SQL-

```
SELECT        alloted.room_no,
              alloted.guest_id,
              uses.facility_id
FROM
              hotel.alloted,
              hotel.uses
WHERE         uses.guest_id = alloted.guest_id AND
    uses.facility_id = 'FC001' AND alloted.room_no = 'A401';
```

Output-

	room_no character varying(5)	guest_id character varying(5)	facility_id character varying(5)
1	A401	C1001	FC001
2	A401	C1007	FC001
3	A401	C1016	FC001
4	A401	C1025	FC001

Figure 47: **Executed output table**

Qu_10- List out the guest_id who were alloted more than or equal to 2 rooms.?

Ans- SQL-

```
SELECT     alloted.guest_id,
           count(alloted.room_no) AS no_of_rooms
FROM          hotel.alloted
GROUP BY      alloted.guest_id
HAVING        count( alloted.room_no ) >= 2
ORDER BY      alloted.guest_id ;
```

Output-

	guest_id character varying(5)	no_of_rooms bigint
1	C1001	2
2	C1002	2
3	C1003	2
4	C1004	2
5	C1005	2
6	C1006	2
7	C1007	2
8	C1008	2
9	C1012	2
10	C1017	2
11	C1021	2
12	C1025	2

Figure 48: **Executed output table**

Qu_11- Total Amount paid by the family type customers?

Ans- SQL-

```
SELECT   sum(bill.amount) AS total_family_amount
FROM     hotel.bill,
         hotel.family
WHERE    family.guest_id = bill.guest_id;
```

Output-

	total_family_amount bigint
1	102245

Figure 49: **Executed output table**

Qu_12- Total Amount paid by the company type customers?

Ans- SQL-

```
SELECT   sum(bill.amount) AS total_company_amount
FROM     hotel.bill,
         hotel.company
WHERE    company.guest_id = bill.guest_id;
```

Output-

	total_company_amount bigint
1	391907

Figure 50: Executed output table

Qu_13- Find out the name of that company that came with maximum number of employee?

Ans- SQL-

```
DROP TABLE company_count;
CREATE TABLE company_count AS ( SELECT company.name,
count(company_members.name) AS no_of_employees
FROM  hotel.company,
      hotel.company_members
WHERE company.guest_id = company_members.guest_id AND
    company_members.cname = company.name GROUP BY company.name
    );
DROP TABLE max_count;
CREATE TABLE max_count AS ( SELECT max(no_of_employees) AS
               no_of_employees FROM company_count );
SELECT *
FROM  max_count NATURAL JOIN company_count;
```

Output-

	no_of_employees bigint	name character varying(40)
1	16	Microsoft

Figure 51: Executed output table

Qu_14- Most Profitable guest_id that is of family type customer?

Ans- SQL-

```
DROP TABLE    family_amount ;
CREATE TABLE     family_amount AS (
SELECT max(bill.amount) AS amount
FROM          hotel.bill,
              hotel.family
WHERE    family.guest_id = bill.guest_id );
SELECT *
FROM     hotel.bill NATURAL JOIN family_amount
```

Output-

	amount integer	bill_no character varying(5)	guest_id character varying(5)	payment_date date	paying_method character varying(40)
1	9860	B1035	F1016	2015-01-15	By-Debit Card

Figure 52: Executed output table

Qu_15- Most Profitable guest_id that is of company type customer?

Ans- SQL-

```
DROP TABLE company_amount ;
CREATE TABLE comnpany_amount AS ( SELECT max(bill.amount)
AS amount
FROM          hotel.bill,
              hotel.company
WHERE    company.guest_id = bill.guest_id );
SELECT *
FROM hotel.bill NATURAL JOIN company_amount
```

Output-

	amount integer	bill_no character varying(5)	guest_id character varying(5)	payment_date date	paying_method character varying(40)
1	39044	B1011	C1006	2015-01-05	By-Cheque

Figure 53: Executed output table

Qu_16- List out the Head_SSN with Head_Name who came twice to the hotel?

Ans- SQL-

```
SELECT DISTINCT    f1.family_head_ssn,
                   f1.family_head_name,
                   count(f1.guest_id)
FROM             hotel.family f1,
                 hotel.family f2
WHERE      f1.family_head_ssn = f2.family_head_ssn AND
            f1.guest_id != f2.guest_id
GROUP BY      f1.family_head_ssn,
              f1.family_head_name
HAVING     count(f1.guest_id) = 2
```

Output-

	family_head_ssn character varying(9)	family_head_name character varying(50)	count bigint
1	222334455	Raman Lamba	2
2	222335566	Yogesh Nagar	2

Figure 54: Executed output table

Qu_17- List the facility used by more than 10 customers?

Ans- SQL-

```
SELECT          uses.facility_id,
              count(uses.guest_id) AS c,
              facility.facility_type
FROM          hotel.facility,
              hotel.uses
WHERE       uses.facility_id = facility.facility_id
GROUP BY    uses.facility_id,
              facility.facility_type
HAVING      count(uses.guest_id) >= 10
```

Output-

	facility_id character varying(5)	c bigint	facility_type character varying(40)
1	FC002	16	WiFi Access
2	FC006	17	Laundry
3	FC004	10	Currency Exchange
4	FC001	34	In-Room Dininig
5	FC008	38	Indoor Parking

Figure 55: Executed output table

Qu_18- List out the different company names with their number of employees?

Ans- SQL-

```
SELECT          company.cname,
              count(company_members.name)
FROM          hotel.company_members,
              hotel.company
WHERE    company_members.guest_id = company.guest_id
GROUP BY    company.cname
```

Output-

	cname character varying(40)	count bigint
1	Hewlett Packard	10
2	Samsung	7
3	Lenovo	5
4	Acer	9
5	Microsoft	16
6	Nestle	11
7	Dell	4
8	Sony	6
9	Colgate	4
10	Toshiba	6
11	Torrent	4
12	Nike Inc	6
13	Intel	5
14	Yahoo	5
15	Honda	5
16	Apple	4
17	Wipro	4
18	Ford	4
19	BMW	4
20	TVS Motors	4
21	IBM	9
22	Volkswagen	7
23	Tata Motors	8

Figure 56: **Executed output table**

Qu_19- List out the guest_id of the family type with the maximum number of members with them?

Ans- SQL-

```
DROP TABLE      f_members;
CREATE TABLE    f_members AS ( SELECT family.guest_id,
                sum(family.no_of_adults family.no_of_children)
                    AS no_of_members
FROM    hotel.family GROUP BY family.guest_id );
DROP TABLE    f1_members;
CREATE TABLE  f1_members AS ( SELECT max(no_of_members) AS
                          no_of_members FROM f_members );
SELECT  *
FROM    f_members NATURAL JOIN f1_members
```

Output-

	no_of_members bigint	guest_id character varying(5)
1	6	F1024

Figure 57: **Executed output table**

Qu_20- List out the facility used by the most number of customers?

Ans- SQL-

```
DROP TABLE    facility_used;
CREATE TABLE  facility_used AS ( SELECT facility.facility_type,
                                    facility.facility_id,
                                    count(uses.guest_id) AS
                                        facility_count
              FROM       hotel.facility,
                         hotel.uses
              WHERE      uses.facility_id = facility.facility_id
GROUP BY          facility.facility_id,
                  facility.facility_type );
DROP TABLE      max_facility_used;
CREATE TABLE    max_facility_used AS ( SELECT
   max(facility_count) AS facility_count FROM  facility_used );
SELECT *
FROM  max_facility_used NATURAL JOIN facility_used
```

Output-

	facility_count bigint	facility_type character varying(40)	facility_id character varying(5)
1	38	Indoor Parking	FC008

Figure 58: Executed output table

Qu_21- Find out the total food amount on a particular date given by the customers?

Ans- SQL-

```
SELECT          orders.date_of_orders,
                sum(food.rate * orders.quantity)
FROM          hotel.orders,
              hotel.food
WHERE         food.product_id = orders.product_id
GROUP BY      orders.date_of_orders
ORDER BY      orders.date_of_orders
```

Output-

	date_of_orders date	sum bigint
1	2015-01-01	6776
2	2015-01-02	3983
3	2015-01-03	8370
4	2015-01-04	12885
5	2015-01-05	8035
6	2015-01-06	2567
7	2015-01-07	3297
8	2015-01-08	5580
9	2015-01-09	5403
10	2015-01-10	10140
11	2015-01-11	11560
12	2015-01-12	10809
13	2015-01-13	13208
14	2015-01-14	9235
15	2015-01-15	21906
16	2015-01-16	14910
17	2015-01-17	5569

Figure 59: Executed output table

Qu_22- List out the guest_id that orders all types of foods?

Ans- SQL-

```
SELECT  *
FROM    hotel.guest AS g
WHERE NOT EXISTS ( ( SELECT   food.type
                     FROM    hotel.food )
EXCEPT ( SELECT       food.type
         FROM       hotel.orders
JOIN hotel.food ON orders.product_id = food.product_id
WHERE orders.guest_id = g.guest_id
 ) );
```

Output-

	guest_id character varying(5)	entry_time timestamp without time zone

Figure 60: **Executed output table**

Qu_23- List out the total number of people who checked out on 11-01-2015 ?

Ans- SQL-

```
SELECT DISTINCT    alloted.check_out_date,
                   alloted.guest_id
FROM               hotel.alloted,
                   hotel.company,
                   hotel.family
WHERE (company.guest_id = allot.guest_id OR family.guest_id =
    allot.guest_id) AND  allot.check_out_date = '2015-01-11';
```

Output-

	check_out_date date	guest_id character varying(5)
1	2015-01-11	F1011
2	2015-01-11	C1012

Figure 61: **Executed output table**

Qu_24- List out the orders for the guest_id - C1004 during his period of living ?

Ans- SQL-

```
SELECT        orders.guest_id,
              orders.product_id,
              orders.date_of_orders,
              orders."time",
              orders.quantity
FROM     hotel.orders
WHERE    orders.guest_id = 'C1004'
```

Output-

	guest_id character varying(5)	product_id character varying(5)	date_of_orders date	time time without time zone	quantity smallint
1	C1004	P0082	2015-01-03	12:30:00	1
2	C1004	P0094	2015-01-03	12:30:00	2
3	C1004	P0099	2015-01-03	12:30:00	2
4	C1004	P0125	2015-01-03	12:30:00	1
5	C1004	P0008	2015-01-03	21:00:00	2
6	C1004	P0013	2015-01-03	21:00:00	1
7	C1004	P0019	2015-01-03	21:00:00	2
8	C1004	P0031	2015-01-04	13:00:00	1
9	C1004	P0038	2015-01-04	13:00:00	1
10	C1004	P0053	2015-01-04	13:00:00	2

Figure 62: **Executed output table**

Qu_25- List out the most frequently alloted room ?

Ans- SQL-

```
DROP TABLE frequency1;
CREATE TABLE frequency1 AS ( SELECT alloted.room_no,
count(alloted.guest_id) as frequency
FROM  hotel.alloted
GROUP BY alloted.room_no
ORDER BY count(allot.guest_id) );
DROP TABLE frequency2;
CREATE TABLE frequency2 AS ( SELECT max(frequency) AS
   frequency  FROM  frequency1 );
SELECT  *
FROM frequency1 NATURAL JOIN frequency2
```

Output-

	frequency bigint	room_no character varying(5)
1	4	A402
2	4	A401

Figure 63: **Executed output table**

Qu_26- Find out the average amount of bill paid by guests visting only for food?

Ans- SQL-

```
DROP TABLE    o_food;
CREATE TABLE o_food AS ( SELECT guest.guest_id FROM
                         hotel.guest
EXCEPT
SELECT  allot.guest_id
FROM  hotel.allot  );
```

```
DROP TABLE o1_food;
CREATE TABLE o1_food AS ( SELECT * FROM hotel.bill NATURAL
    JOIN o_food );
SELECT  sum(o1_food.amount)/count(o1_food.guest_id) AS average
FROM  o1_food;
```

Output-

	average bigint
1	1703

Figure 64: **Executed output table**

Qu_27- Find out the date on which the maximum number customers came to hotel?

Ans- SQL-

```
DROP TABLE new;
CREATE TABLE new AS (
SELECT  count( guest.guest_id) AS no_of_customers,
             date(guest.entry_time)
FROM  hotel.guest
GROUP BY date(guest.entry_time) );
DROP TABLE new1;
CREATE TABLE new1 AS (
SELECT  max(no_of_customers) AS no_of_customers
             FROM  new );
SELECT  *
FROM  new  NATURAL JOIN  new1
```

Output-

	no_of_customers bigint	date date
1	8	2015-01-15

Figure 65: **Executed output table**

Qu_28- List out the guest_id that paid there bills by Cheque?

Ans- SQL-

```
SELECT  bill.guest_id,
        bill.bill_no,
        bill.paying_method
FROM
        hotel.bill
WHERE   bill.paying_method = 'By-Cheque'
```

```
ORDER BY bill.guest_id
```

Output-

	guest_id character varying(5)	bill_no character varying(5)	paying_method character varying(40)
1	C1002	B1006	By-Cheque
2	C1003	B1009	By-Cheque
3	C1004	B1007	By-Cheque
4	C1005	B1010	By-Cheque
5	C1006	B1011	By-Cheque
6	C1007	B1012	By-Cheque
7	C1008	B1013	By-Cheque
8	C1009	B1018	By-Cheque
9	C1010	B1019	By-Cheque
10	C1011	B1026	By-Cheque
11	C1012	B1023	By-Cheque
12	C1013	B1029	By-Cheque
13	C1014	B1027	By-Cheque
14	C1015	B1030	By-Cheque
15	C1016	B1032	By-Cheque
16	C1017	B1034	By-Cheque
17	C1018	B1040	By-Cheque
18	C1019	B1041	By-Cheque
19	C1020	B1042	By-Cheque
20	C1021	B1043	By-Cheque
21	C1022	B1047	By-Cheque
22	C1023	B1048	By-Cheque
23	C1024	B1044	By-Cheque
24	C1025	B1049	By-Cheque

Figure 66: **Executed output table**

Qu_29- List out the guest_id that paid there bills by Cash?

Ans- SQL-

```
SELECT  bill.guest_id,
        bill.bill_no,
        bill.paying_method
FROM
        hotel.bill
WHERE   bill.paying_method = 'By-Cash'
ORDER BY bill.guest_id
```

Output-

	guest_id character varying(5)	bill_no character varying(5)	paying_method character varying(40)
1	C1001	B1004	By-Cash
2	F1001	B1001	By-Cash
3	F1002	B1002	By-Cash
4	F1003	B1003	By-Cash
5	F1007	B1015	By-Cash
6	F1008	B1016	By-Cash
7	F1009	B1017	By-Cash
8	F1010	B1020	By-Cash
9	F1012	B1021	By-Cash
10	F1013	B1022	By-Cash
11	F1014	B1028	By-Cash
12	F1015	B1025	By-Cash
13	F1018	B1031	By-Cash
14	F1019	B1036	By-Cash
15	F1020	B1037	By-Cash
16	F1022	B1038	By-Cash
17	F1023	B1039	By-Cash
18	F1024	B1046	By-Cash

Figure 67: **Executed output table**

Qu_30- List out the guest_id that paid there bills by Debit-Card?

Ans- SQL-

```
SELECT  bill.guest_id,
        bill.bill_no,
        bill.paying_method
FROM
        hotel.bill
WHERE   bill.paying_method = 'By-Debit Card'
ORDER BY bill.guest_id
```

Output-

	guest_id character varying(5)	bill_no character varying(5)	paying_method character varying(40)
1	F1004	B1005	By-Debit Card
2	F1005	B1008	By-Debit Card
3	F1006	B1014	By-Debit Card
4	F1011	B1024	By-Debit Card
5	F1016	B1035	By-Debit Card
6	F1017	B1033	By-Debit Card
7	F1021	B1045	By-Debit Card
8	F1025	B1050	By-Debit Card

Figure 68: **Executed output table**

Qu_31- List name, guest_id of families and company check in on 12-01-2015?

Ans- SQL-

```
SELECT DISTINCT    alloted.guest_id,
                   alloted.check_in_date,
                   family.family_head_ssn AS head_ssn_Cname
FROM               hotel.alloted,
                   hotel.family
WHERE              (family.guest_id = allot.guest_id ) AND
                  allot.check_in_date = '2015-01-12'  UNION
SELECT DISTINCT  allot.guest_id,
                    allot.check_in_date,
                    company.name
FROM                hotel.allot,
                    hotel.company
WHERE              (company.guest_id = alloted.guest_id ) AND
                    alloted.check_in_date = '2015-01-12'
```

Output-

	guest_id character varying(5)	check_in_date date	head_ssn_cname character varying
1	C1016	2015-01-12	Colgate
2	F1016	2015-01-12	222334455
3	C1015	2015-01-12	Dell

Figure 69: **Executed output table**

Qu_32- List out guest id with their room-no. that are of family type?

Ans- SQL-

```
SELECT      family.guest_id,
            alloted.room_no
FROM        hotel.alloted,
            hotel.family
WHERE       family.guest_id = alloted.guest_id;
```

Output-

	guest_id character varying(5)	room_no character varying(5)
1	F1004	A002
2	F1005	A101
3	F1006	A102
4	F1007	A003
5	F1008	A103
6	F1011	A104
7	F1012	A301
8	F1014	A201
9	F1016	A005
10	F1017	A006
11	F1019	A008
12	F1021	A009
13	F1025	A202

Figure 70: **Executed output table**

Qu_33- List out guest id with their room-no. that are of company type?

Ans- SQL-

```
SELECT      company.guest_id,
            alloted.room_no
FROM        hotel.alloted,
            hotel.company
WHERE       company.guest_id = alloted.guest_id;
```

Output-

	guest_id character varying(5)	room_no character varying(5)
1	C1001	A402
2	C1001	A401
3	C1002	A404
4	C1002	A403
5	C1003	A406
6	C1003	A405
7	C1004	A502
8	C1004	A501
9	C1005	A504
10	C1005	A503
11	C1006	A506
12	C1006	A505
13	C1007	A402
14	C1007	A401
15	C1008	A404
16	C1008	A403
17	C1009	A405
18	C1010	A406
19	C1011	A501
20	C1012	A503
21	C1012	A502
22	C1013	A504
23	C1014	A505
24	C1015	A506
25	C1016	A401
26	C1017	A403
27	C1017	A402
28	C1018	A404
29	C1019	A405
30	C1020	A406
31	C1021	A502
32	C1021	A501
33	C1022	A503
34	C1023	A504
35	C1024	A505
36	C1025	A402
37	C1025	A401

Figure 71: **Executed output table**

Qu_34- List out the facility id used by guest that lived in room-no = A101?

Ans- SQL-

```
SELECT      alloted.room_no,
            uses.facility_id,
            facility.facility_type
FROM        hotel.uses,
            hotel.alloted,
            hotel.facility
WHERE    uses.facility_id = facility.facility_id AND
alloted.guest_id = uses.guest_id AND alloted.room_no = 'A101'
```

Output-

	room_no character varying(5)	facility_id character varying(5)	facility_type character varying(40)
1	A101	FC008	Indoor Parking
2	A101	FC005	Babysitting
3	A101	FC006	Laundry
4	A101	FC010	Jiva Spa

Figure 72: **Executed output table**

Qu_35- List out the date on which maximum number of customer of family tpye came to hotel?

Ans- SQL-

```
DROP TABLE new2;
CREATE TABLE new2 AS ( SELECT
      count(guest.guest_id) as count,
      date(guest.entry_time)
FROM     hotel.guest,
         hotel.family
WHERE       family.guest_id = guest.guest_id
GROUP BY    date(guest.entry_time) );
DROP TABLE    new3;
CREATE TABLE  new3 as ( SELECT max(count) as count
                        FROM  new2 );
SELECT      *
FROM  new2  NATURAL JOIN new3
```

Output-

	count bigint	date date
1	4	2015-01-15

Figure 73: **Executed output table**

Qu_36- List out the date on which maximum number of customer of company tpye came to hotel?

Ans- SQL-

```
DROP TABLE new2;
CREATE TABLE new2 AS ( SELECT
      count(guest.guest_id) as count,
      date(guest.entry_time)
FROM     hotel.guest,
         hotel.company
WHERE       company.guest_id = guest.guest_id
GROUP BY   date(guest.entry_time) );
DROP TABLE    new3;
CREATE TABLE  new3 as ( SELECT max(count) as count
                        FROM  new2 );
SELECT     *
FROM  new2  NATURAL JOIN new3
```

Output-

	count bigint	date date
1	4	2015-01-15
2	4	2015-01-04

Figure 74: **Executed output table**

Qu_37- Find out the guest_id that live the maximum number of days in the hotel?

Ans- SQL-

```
DROP TABLE x1;
CREATE TABLE x1 AS( SELECT alloted.guest_id,
   max(alloted.check_out_date-alloted.check_in_date) AS max
FROM   hotel.alloted
GROUP BY alloted.guest_id );
DROP TABLE x2 ;
CREATE TABLE x2 AS ( SELECT max(max) AS max from x1 );
SELECT x1.guest_id,
        x1.max
FROM    x1 natural join x2
```

Output-

	guest_id character varying(5)	max integer
1	F1016	3

Figure 75: **Executed output table**

Qu_38- List names,guest_id of families and company check_in on 11-01-2015 ?

Ans- SQL-

```
SELECT guest_id,Family_Head_name AS name
FROM family
WHERE guest_id IN (SELECT guest_id FROM Guest WHERE
    guest_id like '%F%' and Date(Entry_time)='2015-01-11')
UNION
SELECT guest_id,Name AS name
FROM company
WHERE guest_id in (SELECT guest_id FROM Guest WHERE
    guest_id like '%C%' and Date(Entry_time)='2015-01-11');
```

Output-

	guest_id character varying(5)	name character varying
1	C1013	Honda
2	C1014	Ford
3	F1014	Nirmal Sitharama
4	F1015	Arun Jaitley

Figure 76: **Executed output table**

Qu_39- List names of different companies arrived from 7-1-2015 to 17-1-2015 ?

Ans- SQL-

```
SELECT DISTINCT name
FROM company
WHERE guest_id IN ( SELECT guest_id FROM Allot WHERE
    guest_id like '%F%' and check_in_date >'2015-01-07' and
    check_out_date<='2015-01-15');
```

Output-

	name character varying(40)

Figure 77: **Executed output table**

Qu_40- List guest_id and name of family arrived on 24-1-2015 but not booked room ?

Ans- SQL-

```
SELECT family_members.guest_id,
        family_members.Name
FROM  family,family_members
WHERE family_members.guest_id IN ((SELECT guest_id FROM
    bill WHERE payment_Date='2015-01-24' and guest_id like
    '%F%')
EXCEPT (SELECT Allot.guest_id FROM Allot
        WHERE Allot.guest_id like '%F%' and
            check_in_date='2015-01-24'));
```

Output-

	guest_id character varying(5)	name character varying(40)

Figure 78: **Executed output table**

Embedded SQL in C:

Embedded SQL language help in providing a interface that we can directly insert,update,delete data in the pgAdmin3.

Introduction

We all know that embedded sql is the way to connect our database with the pgAdmin, and we can manually insert, delete and update in the database using the user interface.

Here is an embedded sql code for the insertion, deletion and update for three table of our database:
1. Food
2. Facility &
3. Room

We wanted to mention the attributes in our tables defined above.

1. Food - {Product_Id{PK}, rate, type, name}
2. Facility - {Facility_Id{PK},rate, no_of_hours, facility_type}
3. Room - {Room_No{PK}, rate, type, status_occupied }

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include "sqlca.h"

EXEC SQL INCLUDE sqlca;

EXEC SQL BEGIN DECLARE SECTION;

// Keeping Facility Record
typedef struct
{
varchar f_id[50];
int f_rate;
int f_time;
varchar f_type[50];
} facility_record;

// Keeping Uses Record

typedef struct
{
varchar fa_id[50];
varchar guest_id[50];
int quantity;
} uses_record;

// Keeping Food Record
```

```
typedef struct
{
varchar food_id[50];
int food_rate;
varchar food_type[50];
varchar food_name[50];
} food_record;


// Keeping Room Record
typedef struct
{
varchar room_id[50];
varchar room_type[50];
int room_rate;
varchar room_status[50];
} room_record;

EXEC SQL END DECLARE SECTION;

/////////////////////////////////////////////////////////////////////////////////////////


void prompt(char s[], char t[]) {
char c;
int i = 0;
printf("%s",s);
while ((c = getchar()) != '\n') {
t[i] = c;
i++;
}
t[i] = '\0';
}

// Insert Item on Food

void insert_food()
{

printf("\n Enter details:\n");

EXEC SQL BEGIN DECLARE SECTION;
food_record fo;
EXEC SQL END DECLARE SECTION;

printf("Enter Product ID\n");
scanf("%s",fo.food_id.arr);
fo.food_id.len = strlen(fo.food_id.arr);

printf("Enter the food rate\n");
scanf("%d",&fo.food_rate);
getchar();

printf("Type of the Food\n ");
```

```
scanf("%s",fo.food_type.arr);
fo.food_type.len = strlen(fo.food_type.arr);

printf("Enter the Name of the Food");
scanf("%s",fo.food_name.arr);
fo.food_name.len = strlen(fo.food_name.arr);

EXEC SQL set transaction read write;
EXEC SQL insert into food
    values(:fo.food_id,:fo.food_rate,:fo.food_type,:fo.food_name);

if (sqlca.sqlcode < 0) {
printf("\n\nPRODUCT (%s) already exists\n",fo.food_id.arr);
EXEC SQL rollback work;
EXEC SQL commit;
return;
}
else
{
printf("\n New Food Item Added Succesfully\n");
EXEC SQL commit;
return;
}
}

////////////////////////////////////////////////////////////////////////////////////////

//  Insert Item in Room

void insert_room()
{
printf("Enter the details of the new room\n");
EXEC SQL BEGIN DECLARE SECTION;
room_record ro;
EXEC SQL END DECLARE SECTION;

printf("\nEnter the Room_Number ");
scanf("%s",ro.room_id.arr);
ro.room_id.len=strlen(ro.room_id.arr);

printf("\nEnter the type of the room");
scanf("%s",ro.room_type.arr);
ro.room_type.len=strlen(ro.room_type.arr);

printf("\nEnter the rate of the room");
scanf("%d", &ro.room_rate);
getchar();

printf("\nEnter the Status");
scanf("%s",ro.room_status.arr);
ro.room_status.len=strlen(ro.room_status.arr);


EXEC SQL set transaction read write;
```

```
EXEC SQL insert into room
    values(:ro.room_id,:ro.room_type,:ro.room_rate,:ro.room_status);

if (sqlca.sqlcode < 0) {
printf("\n\nRoom (%s) is already added\n",ro.room_id.arr);
EXEC SQL rollback work;
return;
}
else
{
printf("\n New Room Added Succesfully\n");
EXEC SQL commit;
return;
}
}
////////////////////////////////////////////////////////////////////////////////////////////


void insert_facility()
{
printf("Enter the details of the new facility\n");
EXEC SQL BEGIN DECLARE SECTION;
facility_record fo;
EXEC SQL END DECLARE SECTION;

printf("\nEnter Facility ID ");
scanf("%s",fo.f_id.arr);
fo.f_id.len=strlen(fo.f_id.arr);

printf("\nEnter the rate of the facility");
scanf("%d", &fo.f_rate);
getchar();

printf("\nEnter the no_of_hours of the facility");
scanf("%d", &fo.f_time);
getchar();

printf("\nEnter the type of the facility");
scanf("%s",fo.f_type.arr);
fo.f_type.len=strlen(fo.f_type.arr);

EXEC SQL set transaction read write;
EXEC SQL insert into facility
    values(:fo.f_id,:fo.f_rate,:fo.f_time,:fo.f_type);

if (sqlca.sqlcode < 0) {
printf("\n\nFaclity (%s) is already added\n",fo.f_id.arr);
EXEC SQL rollback work;
return;
}
else
{
printf("\n New Faclity Added Succesfully\n");
EXEC SQL commit;
```

```
return;
}
}
////////////////////////////////////////////////////////////////////////////////////////////

void remove_food()
{
EXEC SQL begin declare section;
food_record frec;
varchar food_n[50];
int onum;
varchar pid[20];
varchar pname[20];
varchar ptype[20];
int prate;
EXEC SQL end declare section;

printf("Enter Product_ID to be deleted: ");
scanf("%s",food_n.arr);
food_n.len = strlen(food_n.arr);


EXEC SQL select *
into :frec
from food
where product_id = :food_n;

if (sqlca.sqlcode > 0) {
printf("Product ID does not exist\n");
return;
}

EXEC SQL declare del_cur cursor for
select product_id from food where product_id = :food_n;

EXEC SQL set transaction read only;
EXEC SQL open del_cur;
EXEC SQL fetch del_cur into :onum;
if (sqlca.sqlcode == 0) {
printf("food_item is in ordered so delete order first\n");
EXEC SQL commit;
return;
}

EXEC SQL commit;
EXEC SQL set transaction read write;
EXEC SQL delete from food where product_id = :food_n;
EXEC SQL DECLARE deleteS CURSOR FOR
           SELECT * FROM food WHERE product_id = :food_n;

   EXEC SQL OPEN deleteS;

   do {
      EXEC SQL FETCH deleteS INTO :pid, :prate, :ptype, :pname;
```

```
      if (sqlca.sqlcode != 0)
       break;
      printf( "\nDeleting the product...\n");
      EXEC SQL DELETE FROM food WHERE CURRENT OF deleteS;
      } while ( 1 );

   EXEC SQL CLOSE deleteS;
printf("\nFood DELETED\n");
EXEC SQL commit;
}

////////////////////////////////////////////////////////////////////////////////////////////////

void remove_facility()
{
EXEC SQL begin declare section;
facility_record farec;
uses_record urec;
varchar faci_n[50];
int onum;
varchar fid[20];
varchar ftype[20];
int ftime[20];
int frate;
EXEC SQL end declare section;

printf("Enter Facility_ID to be deleted: ");
scanf("%s",faci_n.arr);
faci_n.len = strlen(faci_n.arr);


EXEC SQL select *
into :farec
from facility
where facility_id = :faci_n;

if (sqlca.sqlcode > 0) {
printf("Facility ID does not exist\n");
   return;

}


EXEC SQL declare del_cur1 cursor for
select facility_id from facility where facility_id = :faci_n;

EXEC SQL set transaction read only;
EXEC SQL open del_cur1;
EXEC SQL fetch del_cur1 into :onum;
if (sqlca.sqlcode == 0) {
printf("facility_id is in use so delete order first\n");
EXEC SQL commit;
return;
}
```

```
EXEC SQL commit;
EXEC SQL set transaction read write;
EXEC SQL delete from facility where facility_id = :faci_n;
EXEC SQL DECLARE deleteP CURSOR FOR
          SELECT * FROM facility WHERE facility_id = :faci_n;

   EXEC SQL OPEN deleteP;

   do {
      EXEC SQL FETCH deleteP INTO :fid, :frate, :ftime, :ftype;
      if (sqlca.sqlcode != 0) break;
      printf( "\nDeleting the facility id...\n");
      EXEC SQL DELETE FROM facility WHERE CURRENT OF deleteS;
      } while ( 1 );

   EXEC SQL CLOSE deleteP;
printf("\nFacility DELETED\n");
EXEC SQL commit;
}
////////////////////////////////////////////////////////////////////////////////////////////

void remove_room()
{
EXEC SQL begin declare section;
room_record rrec;
varchar room_n[50];
int onum;
varchar rid[20];
varchar rtype[20];
varchar rstatus[20];
int rrate;
EXEC SQL end declare section;

printf("Enter Room_Number to be deleted: ");
scanf("%s",room_n.arr);
room_n.len = strlen(room_n.arr);


EXEC SQL select *
into :rrec
from room
where room_no = :room_n;

if (sqlca.sqlcode > 0) {
printf("Room does not exist\n");
return;
}


EXEC SQL declare del_cur2 cursor for
select room_no from room where room_no = :room_n;

EXEC SQL set transaction read only;
```

```
EXEC SQL open del_cur2;
EXEC SQL fetch del_cur2 into :onum;
if (sqlca.sqlcode == 0) {
printf("room_no is alloted first delete the allot\n");
EXEC SQL commit;
return;
}

EXEC SQL commit;
EXEC SQL set transaction read write;
EXEC SQL delete from room where room_no = :room_n;
EXEC SQL DECLARE deleteQ CURSOR FOR
          SELECT * FROM room facility WHERE room_no= :room_n;

   EXEC SQL OPEN deleteQ;

   do {
      EXEC SQL FETCH deleteQ INTO :rid, :rtype, :rrate, :rstatus;
      if (sqlca.sqlcode != 0) break;
      printf( "\nDeleting the given Room...\n");
      EXEC SQL DELETE FROM room WHERE CURRENT OF deleteS;
      } while ( 1 );

   EXEC SQL CLOSE deleteQ;
printf("\nRoom DELETED\n");
EXEC SQL commit;
}

////////////////////////////////////////////////////////////////////////////////////////////////////


void update_food() {

EXEC SQL SET SEARCH_PATH TO hotel;
EXEC SQL begin declare section;
food_record food1;
varchar fo_id[40];
varchar foo_id[40];
varchar n[30];
varchar t[30];
int r;
int temp;
EXEC SQL end declare section;

printf("Enter the product_id you want to update\n ");
scanf("%s",fo_id.arr);
fo_id.len = strlen(fo_id.arr);
EXEC SQL select *
into
:food1
from
food
where product_id = :fo_id;
if (sqlca.sqlcode > 0) {
```

```
printf("food_item(%s) does not exist\n",fo_id.arr);
return;
}

printf("Current rate: %d\n",food1.food_rate);
printf("Enter the rate :\n ");
scanf("%d", &r);
food1.food_rate = r;

printf("Current Tpye%s\n",food1.food_type.arr);
printf("New Type of the food : \n");
scanf("%s",t.arr);
if (strlen(t.arr) > 1) {
strcpy(food1.food_type.arr,t.arr);
food1.food_type.len = strlen(food1.food_type.arr);
}

printf("Current Name%s\n",food1.food_name.arr);
printf("New Name of the food :\n ");
scanf("%s", n.arr);
if (strlen(n.arr) > 1) {
strcpy(food1.food_name.arr,n.arr);
food1.food_name.len = strlen(food1.food_name.arr);
}



EXEC SQL set transaction read write;
EXEC SQL UPDATE food
set product_id = :food1.food_id,
rate = :food1.food_rate,
type = :food1.food_type,
name = :food1.food_name
where product_id = :food1.food_id;
if (sqlca.sqlcode < 0) {
printf("\n\nError on Update\n");
EXEC SQL rollback work;
return;
}
EXEC SQL commit;
printf("\nFood Item (%s) updated.\n",food1.food_id.arr);
}

////////////////////////////////////////////////////////////////////////////////////////////////////

void update_room() {

EXEC SQL SET SEARCH_PATH TO hotel;
EXEC SQL begin declare section;
room_record room1;
varchar ro_id[40];
varchar roo_id[40];
varchar rs[30];
varchar t[30];
```

```
int r;
EXEC SQL end declare section;

printf("Enter the room_id you want to update \n");
scanf("%s",ro_id.arr);
ro_id.len = strlen(ro_id.arr);
EXEC SQL select *
into
:room1
from
room
where room_no = :ro_id;
if (sqlca.sqlcode > 0) {
printf("room(%s) does not exist\n",ro_id.arr);
return;
}

printf("Current rate: %d\n",room1.room_rate);
printf("Enter the rate:\n ");
scanf("%d", &r);
room1.room_rate = r;

printf("Current Tpye%s\n",room1.room_type.arr);
printf("New Type of the room:\n");
scanf("%s",t.arr);
if (strlen(t.arr) > 1) {
strcpy(room1.room_type.arr,t.arr);
room1.room_type.len = strlen(room1.room_type.arr);
}

printf("Current Status%s\n",room1.room_status.arr);
printf("New Status of the room: \n");
scanf("%s", rs.arr);
if (strlen(rs.arr) > 1) {
strcpy(room1.room_status.arr,rs.arr);
room1.room_status.len = strlen(room1.room_status.arr);
}



EXEC SQL set transaction read write;
EXEC SQL UPDATE room
set room_no = :room1.room_id,
type = :room1.room_type,
rate = :room1.room_rate,
status_occupied = :room1.room_status
where room_no = :room1.room_id;
if (sqlca.sqlcode < 0) {
printf("\n\nError on Update\n");
EXEC SQL rollback work;
return;
}
EXEC SQL commit;
printf("\nRoom is (%s) updated.\n",room1.room_id.arr);
```

```
}

////////////////////////////////////////////////////////////////////////////////////////////////////

void update_facility() {

EXEC SQL SET SEARCH_PATH TO hotel;
EXEC SQL begin declare section;
facility_record fac1;
varchar fa_id[40];
varchar faa_id[40];
varchar ft[30];
int fh;
int r;
EXEC SQL end declare section;

printf("Enter the facility_id you want to update \n");
scanf("%s",fa_id.arr);
fa_id.len = strlen(fa_id.arr);
EXEC SQL select *
into
:fac1
from
facility
where facility_id = :fa_id;
if (sqlca.sqlcode > 0) {
printf("facility(%s) does not exist\n",fa_id.arr);
return;
}

printf("Current rate: of the facility %d\n",fac1.f_rate);
printf("Enter the rate:\n ");
scanf("%d", &r);
fac1.f_rate = r;

printf("Current Time for the facility%d\n",fac1.f_time);
printf("New Time for the facility\n");
scanf("%d",&fh);
fac1.f_time = fh;


printf("Current Type%s\n",fac1.f_type.arr);
printf("New Type of the facility: \n");
scanf("%s", ft.arr);
if (strlen(ft.arr) > 1) {
strcpy(fac1.f_type.arr,ft.arr);
fac1.f_type.len = strlen(fac1.f_type.arr);
}


EXEC SQL set transaction read write;
EXEC SQL UPDATE facility
set facility_id = :fac1.f_id,
rate = :fac1.f_rate,
```

```
no_of_hours = :fac1.f_time,
facility_type = :fac1.f_type
where facility_id = :fac1.f_id;
if (sqlca.sqlcode < 0) {
printf("\n\nError on Update\n");
EXEC SQL rollback work;
return;
}
EXEC SQL commit;
printf("\nfacility is (%s) updated.\n",fac1.f_id.arr);
}



////////////////////////////////////////////////////////////////////////////////////////////////////

void main_menu()
{
printf("\n\n");
printf("\t\t\t\t\t\tWelcome to GUI For Our Project\n\n");
printf("\t\t\t* *   * ***** *    **** ****  *     * *****   *** ****   *
    * **** *** **** * \n");
printf("\t\t\t * * * * ***  *    *    * *   * * * * ***      *   * *
    **** * *  *  *** *   \n");
printf("\t\t\t *  *   ***** **** **** ****  *   * *  *****    *   ****   *
    * **** *  **** *** \n\n");


printf("\t\t\t\t\t----------------------Hotel Management
    System-----------------\n\n");
printf("\n\t\t\t\t\t\tPlease select an option:\n");
printf("\t\t\t\t\t1. Insert a new Entry\n");
printf("\t\t\t\t\t2. Update an existing entry\n");
printf("\t\t\t\t\t3. Remove an existing Entry\n");
printf("\t\t\t\t\t4. Exit\n\n");
printf("\t\t\t\t\t************************************************************\n");
}

////////////////////////////////////////////////////////////////////////////////////////////////////

void main_insert()
{
int insert_choice;
printf("Enter the type of a new Entry\n");
printf("1. Food\n");
printf("2. Room\n");
printf("3. Facility\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d",&insert_choice);
switch(insert_choice)
{
case 1:
insert_food();
```

```
break;
case 2:
insert_room();
break;
case 3:
insert_facility();
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
}
}

////////////////////////////////////////////////////////////////////////////////////////////////


void main_remove()
{
int remove_choice;
printf("Enter the type of entry you want to delete\n");
printf("1. Food\n");
printf("2. Facility\n");
printf("3. Room\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d",&remove_choice);
switch(remove_choice)
{
case 1:
remove_food();
break;
case 2:
remove_facility();
break;
case 3:
remove_room();
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
}
}


////////////////////////////////////////////////////////////////////////////////////////////////

void main_update()
{
int remove_choice;
printf("Enter the type of entry you want to update\n");
printf("1. Food\n");
printf("2. Facility\n");
```

```
printf("3. Room\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d",&remove_choice);
switch(remove_choice)
{
case 1:
update_food();
break;
case 2:
update_facility();
break;
case 3:
update_room();
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
}
}

////////////////////////////////////////////////////////////////////////////////////////////

int main()
{
EXEC SQL CONNECT TO try_project@localhost:5432 USER postgres;
EXEC SQL set search_path to hotel;;
int main_choice;
X:
main_menu();
printf("\nEnter your choice (eg. 1 for 1st choice): ");
scanf("%d",&main_choice);
switch(main_choice)
{
case 1:
main_insert();
goto X;
break;
case 2:
main_update();
goto X;
break;
case 3:
main_remove();
goto X;
break;
case 4:
EXEC SQL COMMIT;
EXEC SQL DISCONNECT;
exit(0);
default:
printf("Re-enter your choice");
goto X;
```

```
}
EXEC SQL DISCONNECT try_project;
return 0;
}
```

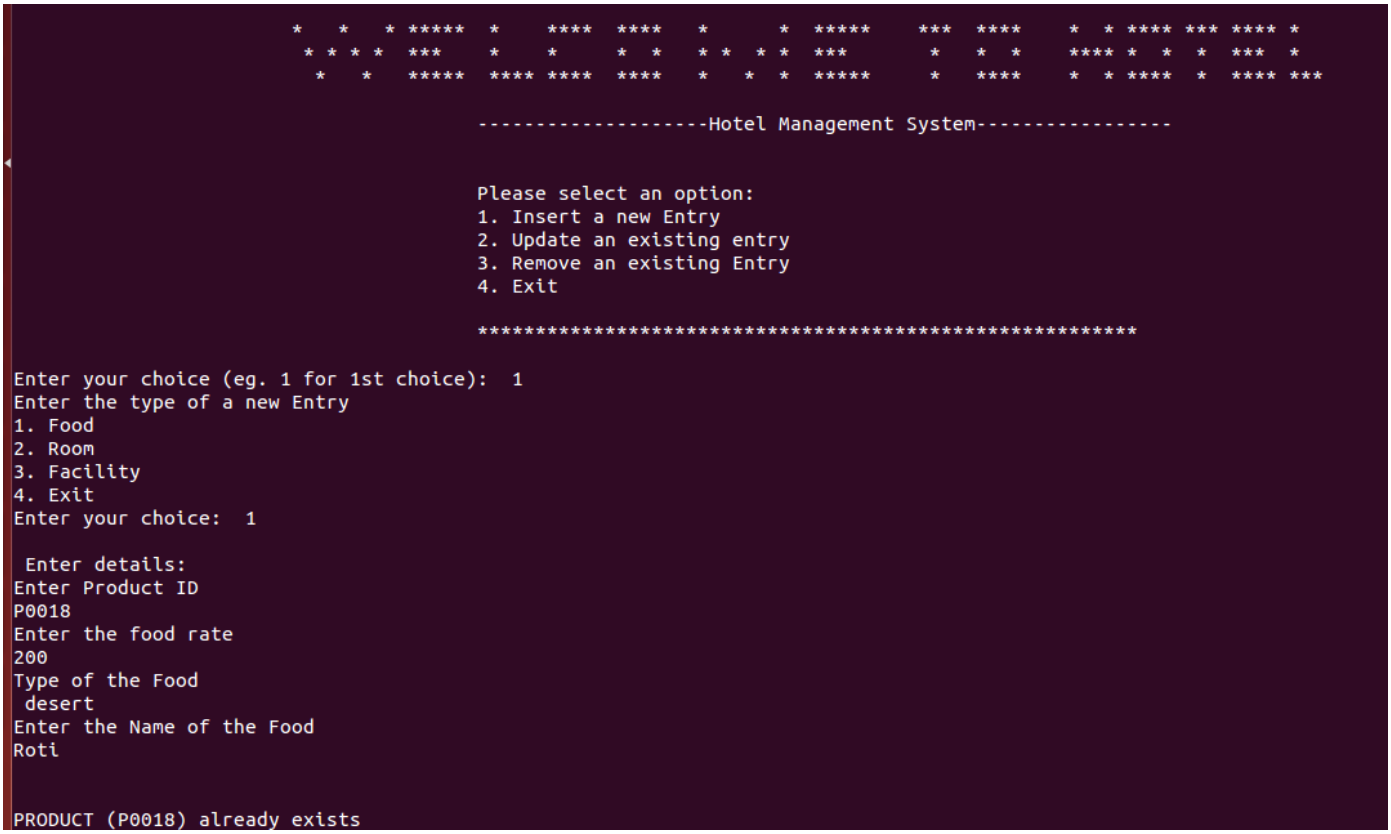

Figure 79: **Sample output of Embedded sql Query**

Conclusion:

HMS resolves almost all queries that a real world HMS should answer. It not only stores a huge database, but also manipulates, deletes, and updates data efficiently.

Refrences :

http://tinman.cs.gsu.edu/~raj/books/Oracle9-chapter-3.pdf
en.wikipedia.org