NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

DBMS

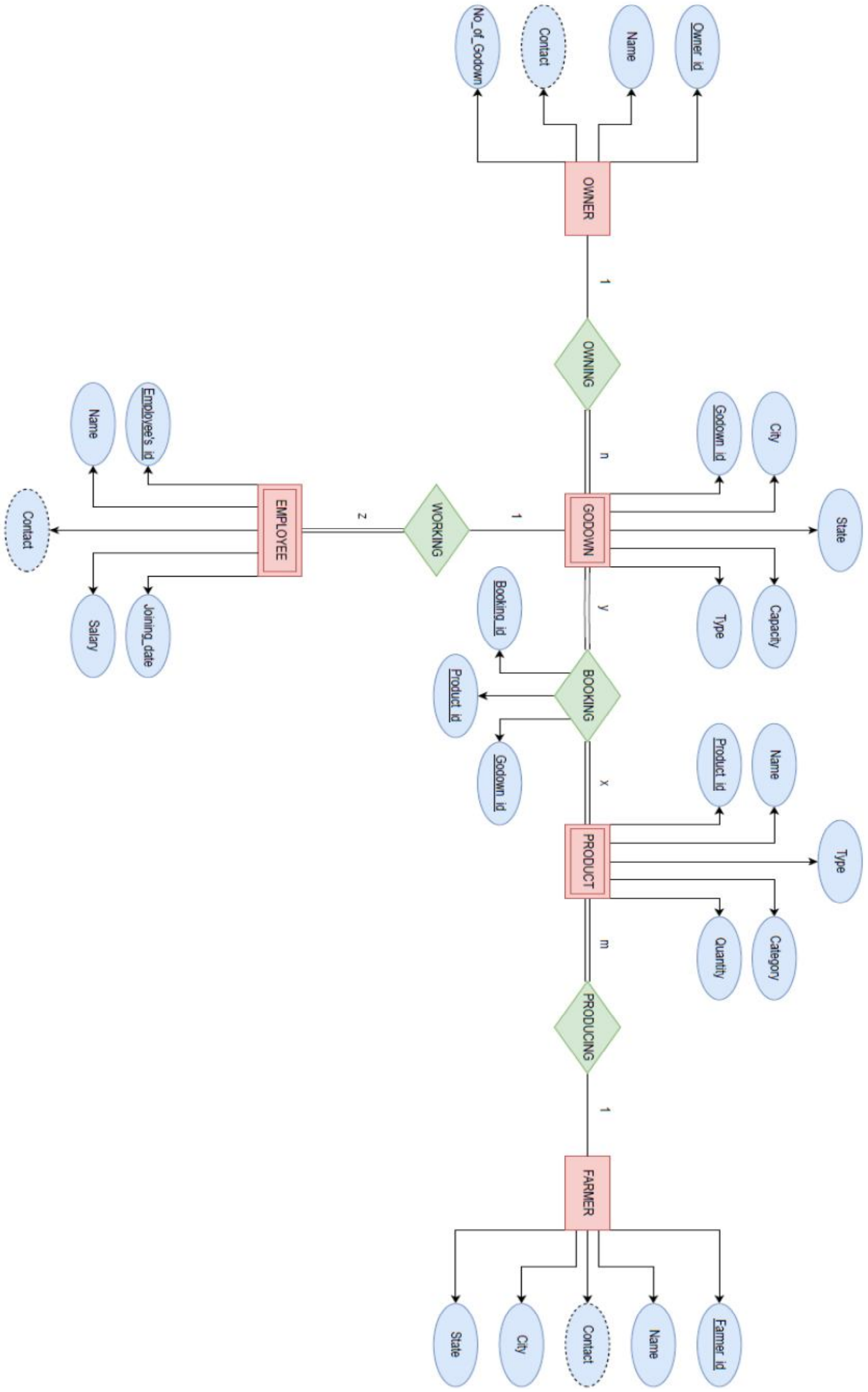
INNOVATIVE ASSIGNMENT

GODOWN MANAGEMENT SYSTEM

Name :Shwet Kheni
Batch:F-1
Roll Number:22BCE337

Name :Shrey Vyas
Batch:F-1
Roll Number:22BCE335

ER Diagram


OWNER
Owner_id
Name
Contact
No_of_Godown
1
OWNING
n
GODOWN
Godown_id
City
State
Capacity
Type
1
WORKING
z
EMPLOYEE
Employee's_id
Name
Contact
Salary
Joining_date
y
BOOKING
Booking_id
Product_id
Godown_id
x
PRODUCT
Product_id
Name
Type
Category
Quantity
m
PRODUCING
1
FARMER
Farmer_id
Name
Contact
City
State

Create Queries

```
create table owner(
    Owner_id int(4)  not null primary key auto_increment,
    O_Name varchar(20),
    O_Contact int(10),
    No_of_godown int(5)
);
```

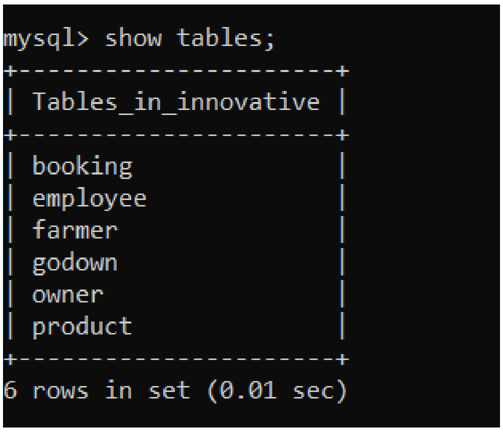
```
create table Godown(
    Godown_id int(4)  not null primary key auto_increment,
    G_city varchar(15),
    G_state varchar(15),
    capacity int(10),
    G_type varchar(5),
    Owner_id int(4) references owner (Owner_id)
);
```

```
create table employee(
    Employee_id int(4) not null primary key auto_increment,
    E_Name varchar(20),
    E_Contact int(10),
    Salary int(15),
    joining_date date,
    Working_Godown int(4)  references Godown(Godown_id)
);
```

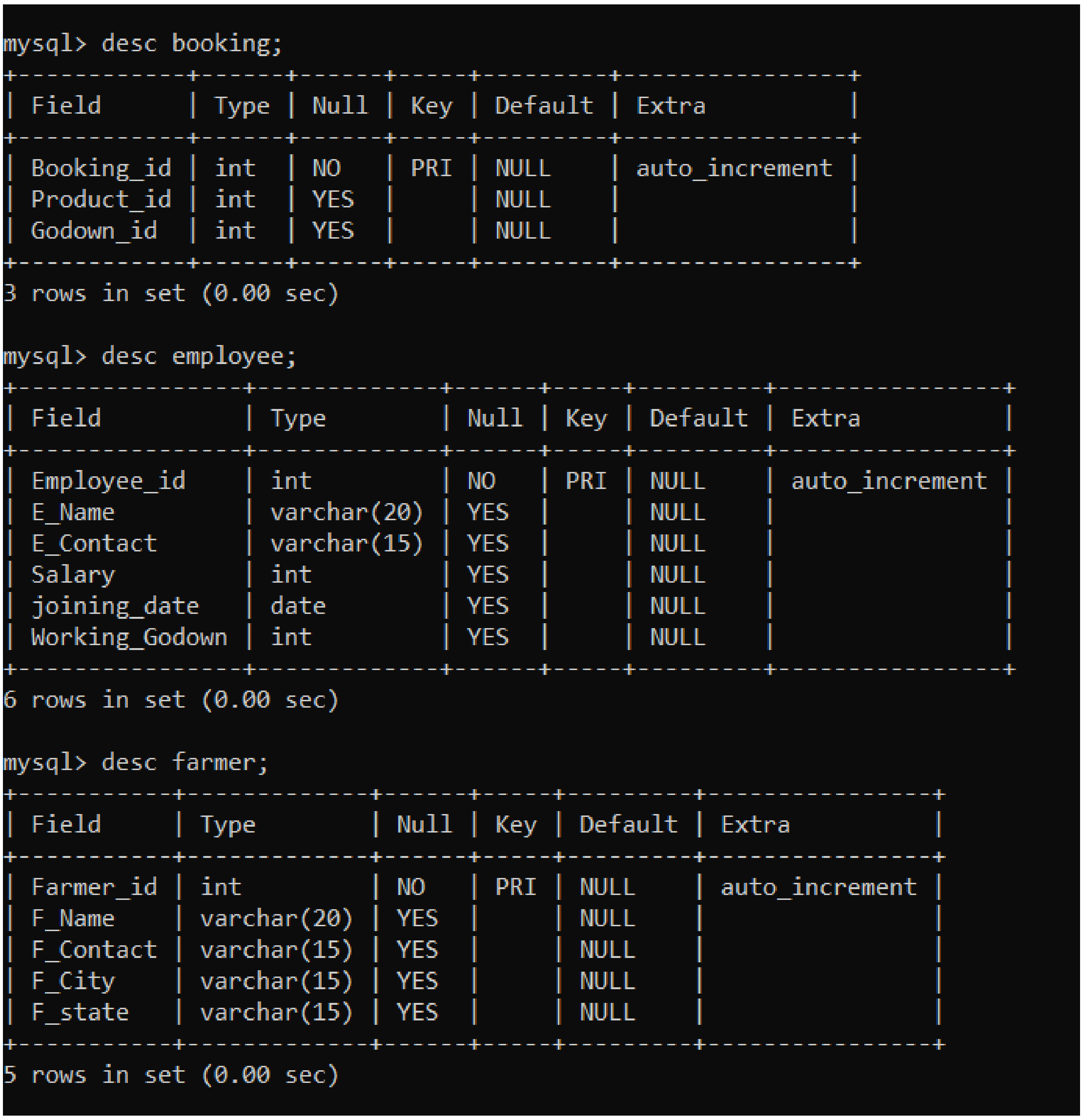
```
create table Farmer(
    Farmer_id int(4) not null primary key auto_increment,
    F_Name varchar(20),
    F_Contact int(10),
    F_City varchar(15),
    F_state varchar(15)
);
```

```
create table Product(
    Product_id int(4) not null primary key auto_increment,
    P_Name varchar(20),
    P_type varchar(22),
    quantity int(10),
    producer int(4)  references Farmer(Farmer_id)
);
```

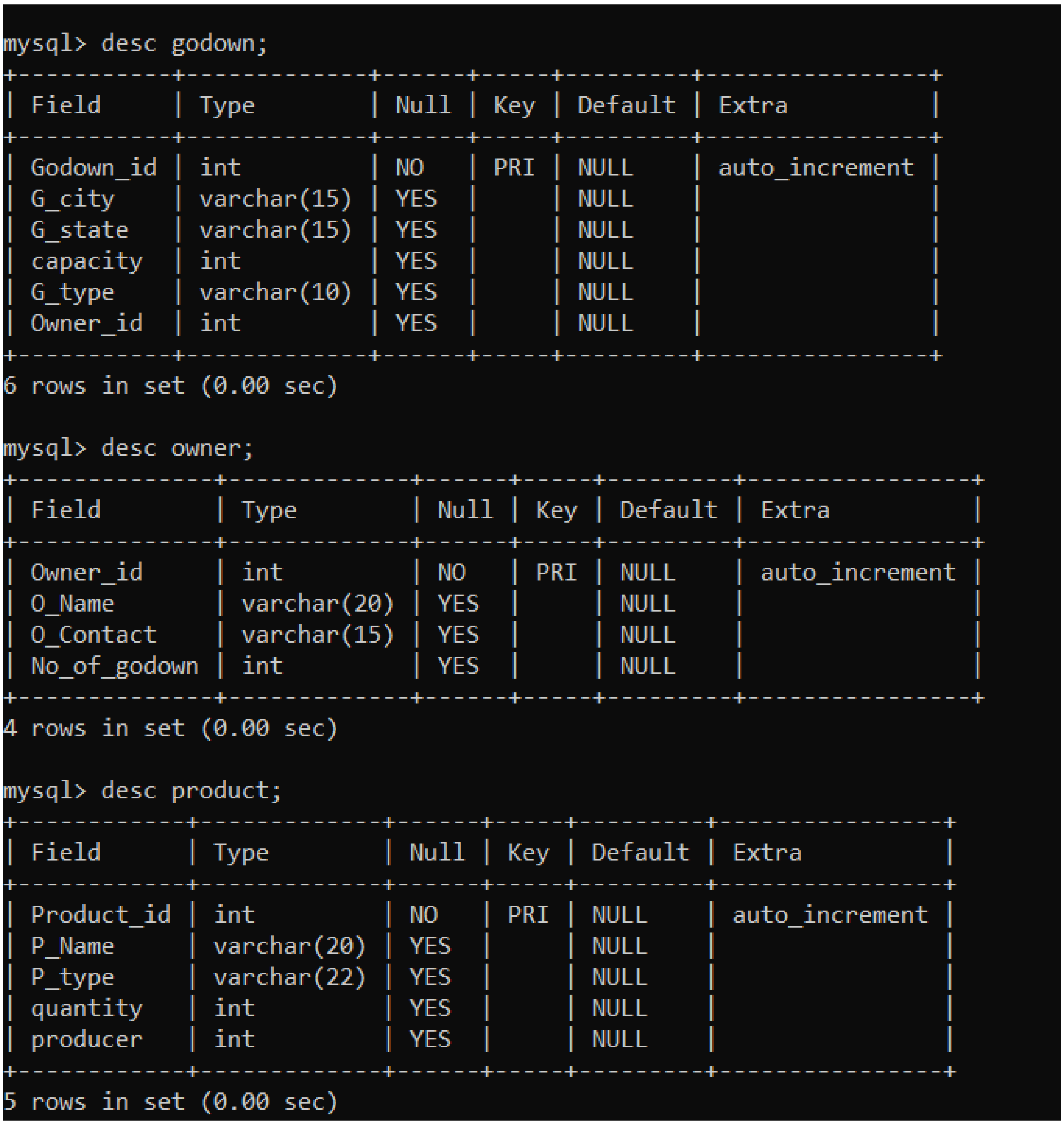
```
create table Booking(
    Booking_id int(4) not null primary key auto_increment,
    Product_id int(4) references  product(Product_id),
    Godown_id int(4) references  Godown(Godown_id)
);
```



```
mysql> show tables;
+----------------------+
| Tables_in_innovative |
+----------------------+
| booking              |
| employee             |
| farmer               |
| godown               |
| owner                |
| product              |
+----------------------+
6 rows in set (0.01 sec)
```

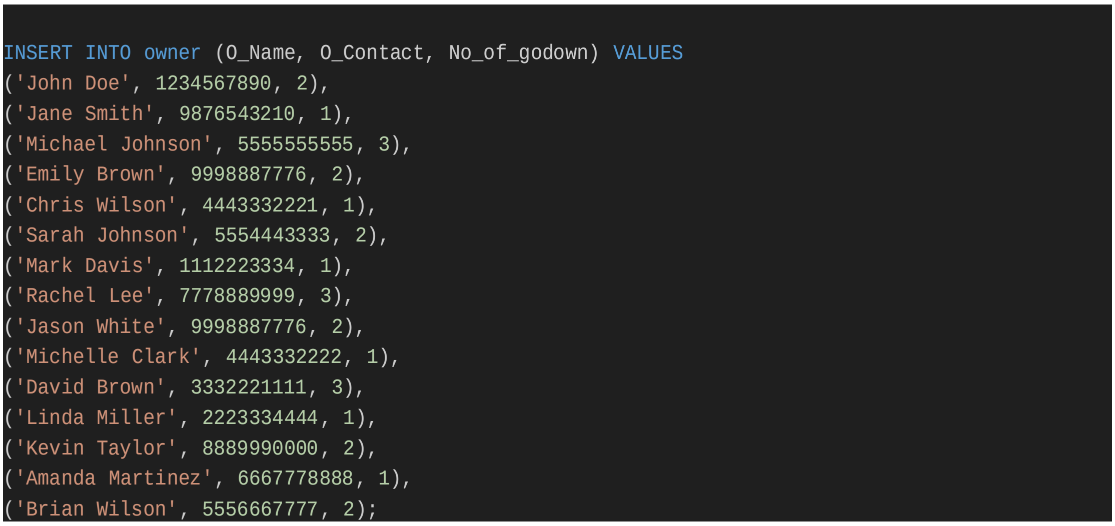


```
mysql> desc booking;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| Booking_id | int  | NO   | PRI | NULL    | auto_increment |
| Product_id | int  | YES  |     | NULL    |                |
| Godown_id  | int  | YES  |     | NULL    |                |
+------------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> desc employee;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| Employee_id    | int         | NO   | PRI | NULL    | auto_increment |
| E_Name         | varchar(20) | YES  |     | NULL    |                |
| E_Contact      | varchar(15) | YES  |     | NULL    |                |
| Salary         | int         | YES  |     | NULL    |                |
| joining_date   | date        | YES  |     | NULL    |                |
| Working_Godown | int         | YES  |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql> desc farmer;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| Farmer_id | int         | NO   | PRI | NULL    | auto_increment |
| F_Name    | varchar(20) | YES  |     | NULL    |                |
| F_Contact | varchar(15) | YES  |     | NULL    |                |
| F_City    | varchar(15) | YES  |     | NULL    |                |
| F_state   | varchar(15) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

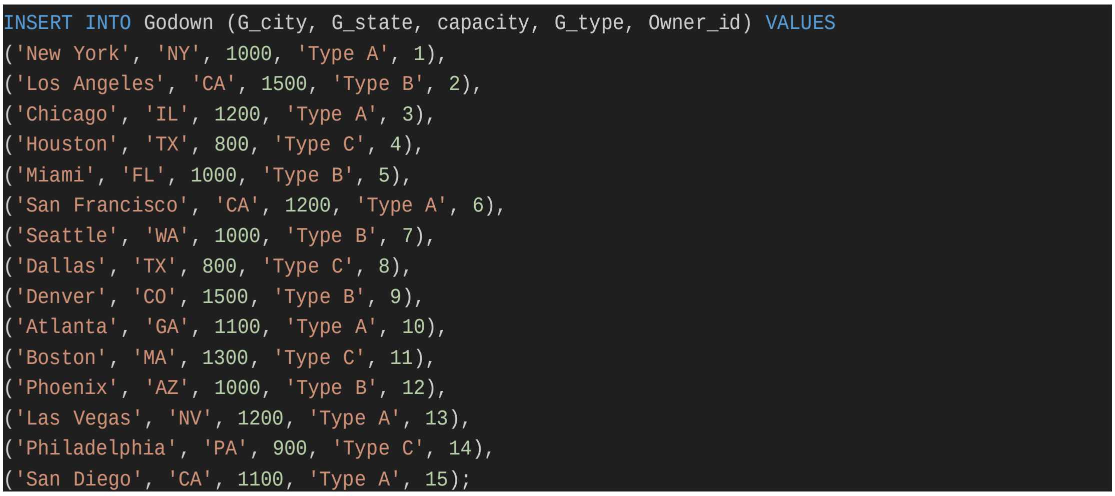


```
mysql> desc godown;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| Godown_id | int         | NO   | PRI | NULL    | auto_increment |
| G_city    | varchar(15) | YES  |     | NULL    |                |
| G_state   | varchar(15) | YES  |     | NULL    |                |
| capacity  | int         | YES  |     | NULL    |                |
| G_type    | varchar(10) | YES  |     | NULL    |                |
| Owner_id  | int         | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)

mysql> desc owner;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| Owner_id     | int         | NO   | PRI | NULL    | auto_increment |
| O_Name       | varchar(20) | YES  |     | NULL    |                |
| O_Contact    | varchar(15) | YES  |     | NULL    |                |
| No_of_godown | int         | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> desc product;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| Product_id | int         | NO   | PRI | NULL    | auto_increment |
| P_Name     | varchar(20) | YES  |     | NULL    |                |
| P_type     | varchar(22) | YES  |     | NULL    |                |
| quantity   | int         | YES  |     | NULL    |                |
| producer   | int         | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

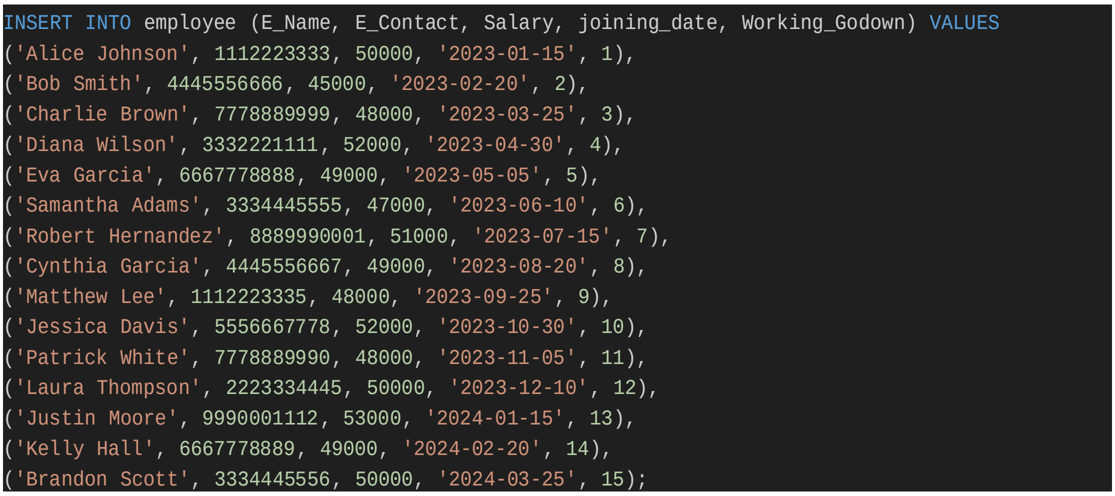
Insert Queries



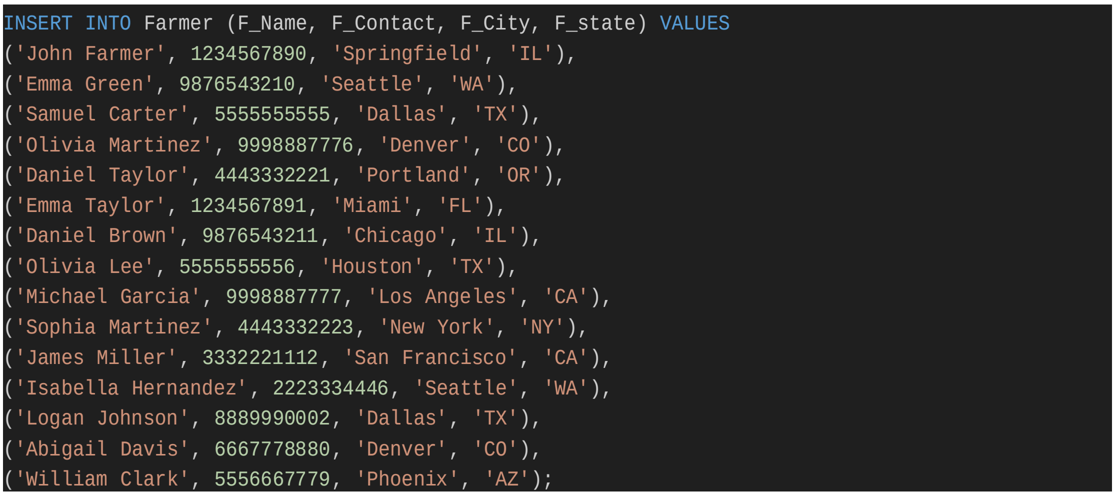
```
INSERT INTO owner (O_Name, O_Contact, No_of_godown) VALUES
('John Doe', 1234567890, 2),
('Jane Smith', 9876543210, 1),
('Michael Johnson', 5555555555, 3),
('Emily Brown', 9998887776, 2),
('Chris Wilson', 4443332221, 1),
('Sarah Johnson', 5554443333, 2),
('Mark Davis', 1112223334, 1),
('Rachel Lee', 7778889999, 3),
('Jason White', 9998887776, 2),
('Michelle Clark', 4443332222, 1),
('David Brown', 3332221111, 3),
('Linda Miller', 2223334444, 1),
('Kevin Taylor', 8889990000, 2),
('Amanda Martinez', 6667778888, 1),
('Brian Wilson', 5556667777, 2);
```



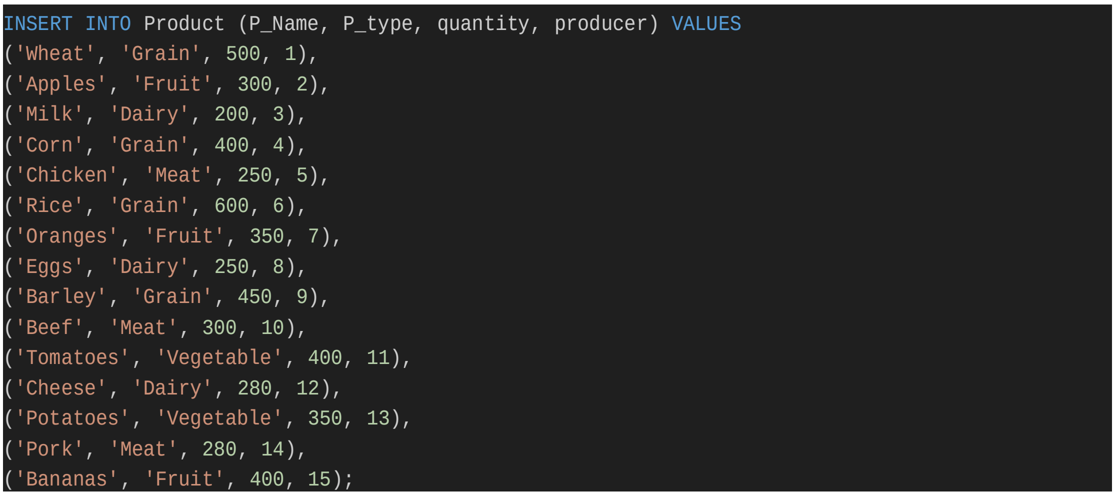
```
INSERT INTO Godown (G_city, G_state, capacity, G_type, Owner_id) VALUES
('New York', 'NY', 1000, 'Type A', 1),
('Los Angeles', 'CA', 1500, 'Type B', 2),
('Chicago', 'IL', 1200, 'Type A', 3),
('Houston', 'TX', 800, 'Type C', 4),
('Miami', 'FL', 1000, 'Type B', 5),
('San Francisco', 'CA', 1200, 'Type A', 6),
('Seattle', 'WA', 1000, 'Type B', 7),
('Dallas', 'TX', 800, 'Type C', 8),
('Denver', 'CO', 1500, 'Type B', 9),
('Atlanta', 'GA', 1100, 'Type A', 10),
('Boston', 'MA', 1300, 'Type C', 11),
('Phoenix', 'AZ', 1000, 'Type B', 12),
('Las Vegas', 'NV', 1200, 'Type A', 13),
('Philadelphia', 'PA', 900, 'Type C', 14),
('San Diego', 'CA', 1100, 'Type A', 15);
```



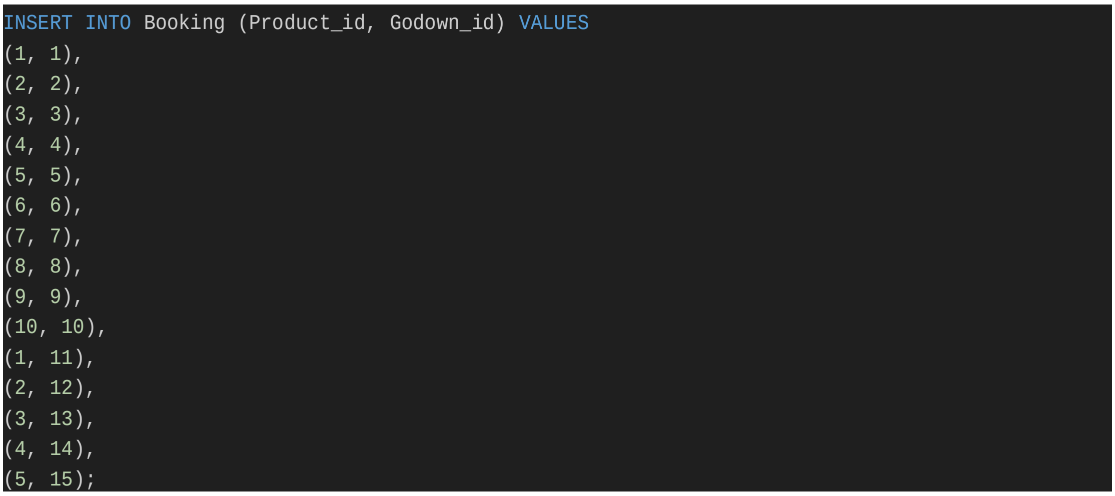
```
INSERT INTO employee (E_Name, E_Contact, Salary, joining_date, Working_Godown) VALUES
('Alice Johnson', 1112223333, 50000, '2023-01-15', 1),
('Bob Smith', 4445556666, 45000, '2023-02-20', 2),
('Charlie Brown', 7778889999, 48000, '2023-03-25', 3),
('Diana Wilson', 3332221111, 52000, '2023-04-30', 4),
('Eva Garcia', 6667778888, 49000, '2023-05-05', 5),
('Samantha Adams', 3334445555, 47000, '2023-06-10', 6),
('Robert Hernandez', 8889990001, 51000, '2023-07-15', 7),
('Cynthia Garcia', 4445556667, 49000, '2023-08-20', 8),
('Matthew Lee', 1112223335, 48000, '2023-09-25', 9),
('Jessica Davis', 5556667778, 52000, '2023-10-30', 10),
('Patrick White', 7778889990, 48000, '2023-11-05', 11),
('Laura Thompson', 2223334445, 50000, '2023-12-10', 12),
('Justin Moore', 9990001112, 53000, '2024-01-15', 13),
('Kelly Hall', 6667778889, 49000, '2024-02-20', 14),
('Brandon Scott', 3334445556, 50000, '2024-03-25', 15);
```



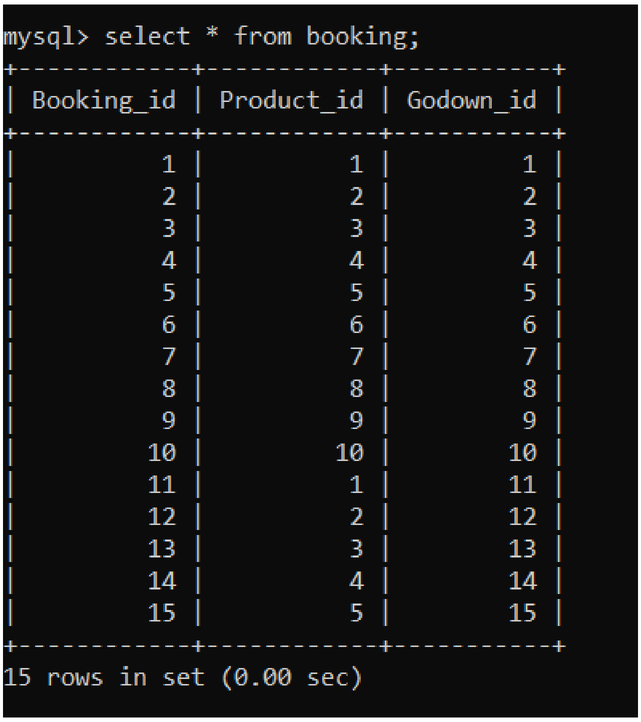
```sql
INSERT INTO Farmer (F_Name, F_Contact, F_City, F_state) VALUES
('John Farmer', 1234567890, 'Springfield', 'IL'),
('Emma Green', 9876543210, 'Seattle', 'WA'),
('Samuel Carter', 5555555555, 'Dallas', 'TX'),
('Olivia Martinez', 9998887776, 'Denver', 'CO'),
('Daniel Taylor', 4443332221, 'Portland', 'OR'),
('Emma Taylor', 1234567891, 'Miami', 'FL'),
('Daniel Brown', 9876543211, 'Chicago', 'IL'),
('Olivia Lee', 5555555556, 'Houston', 'TX'),
('Michael Garcia', 9998887777, 'Los Angeles', 'CA'),
('Sophia Martinez', 4443332223, 'New York', 'NY'),
('James Miller', 3332221112, 'San Francisco', 'CA'),
('Isabella Hernandez', 2223334446, 'Seattle', 'WA'),
('Logan Johnson', 8889990002, 'Dallas', 'TX'),
('Abigail Davis', 6667778880, 'Denver', 'CO'),
('William Clark', 5556667779, 'Phoenix', 'AZ');
```



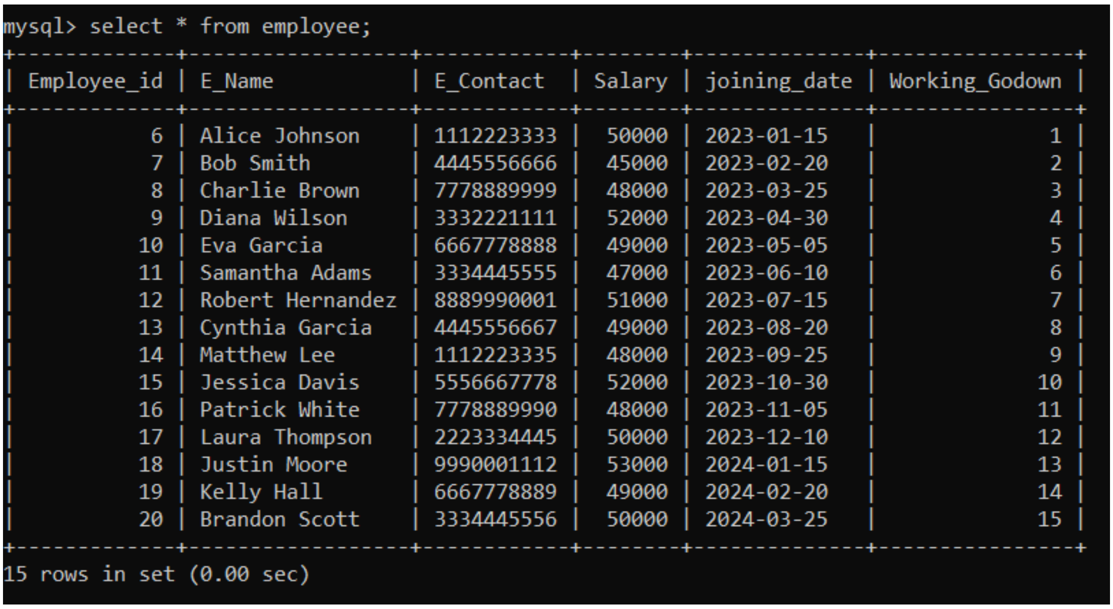
```sql
INSERT INTO Product (P_Name, P_type, quantity, producer) VALUES
('Wheat', 'Grain', 500, 1),
('Apples', 'Fruit', 300, 2),
('Milk', 'Dairy', 200, 3),
('Corn', 'Grain', 400, 4),
('Chicken', 'Meat', 250, 5),
('Rice', 'Grain', 600, 6),
('Oranges', 'Fruit', 350, 7),
('Eggs', 'Dairy', 250, 8),
('Barley', 'Grain', 450, 9),
('Beef', 'Meat', 300, 10),
('Tomatoes', 'Vegetable', 400, 11),
('Cheese', 'Dairy', 280, 12),
('Potatoes', 'Vegetable', 350, 13),
('Pork', 'Meat', 280, 14),
('Bananas', 'Fruit', 400, 15);
```



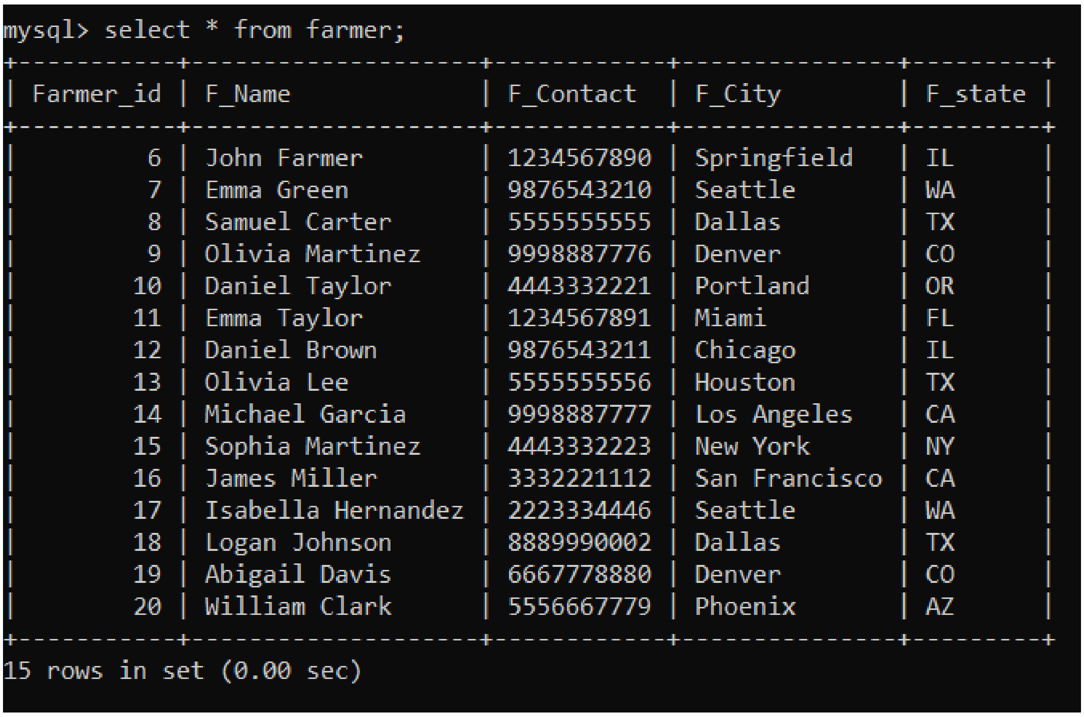
```sql
INSERT INTO Booking (Product_id, Godown_id) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10),
(1, 11),
(2, 12),
(3, 13),
(4, 14),
(5, 15);
```



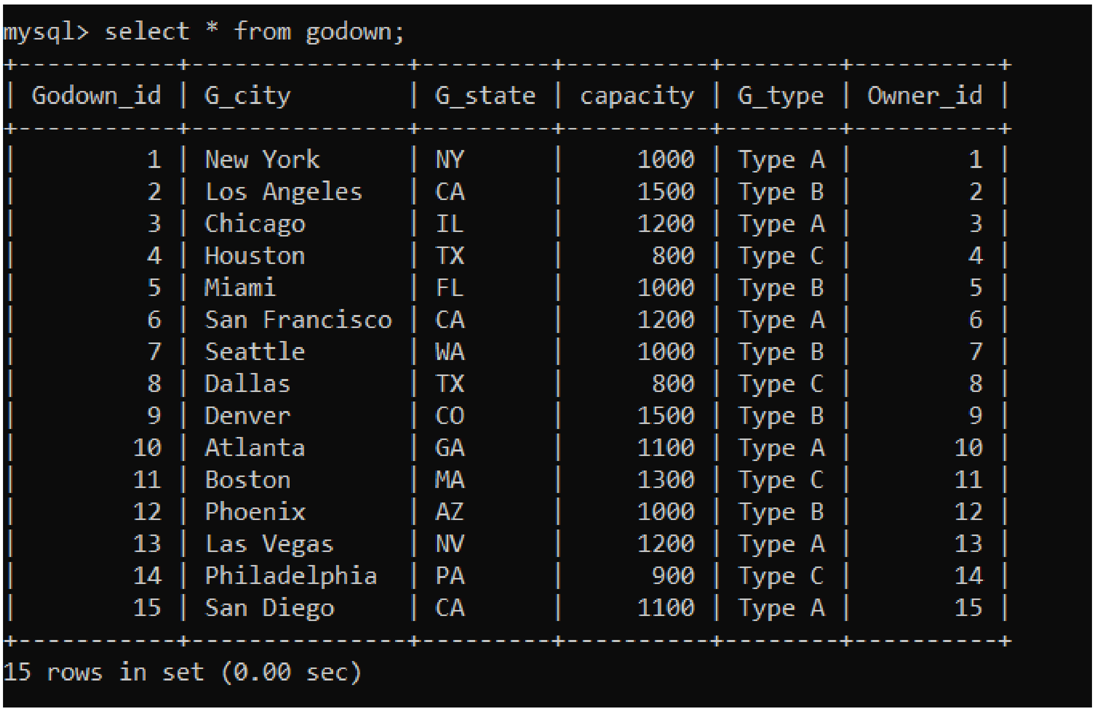
```
mysql> select * from booking;
+------------+------------+-----------+
| Booking_id | Product_id | Godown_id |
+------------+------------+-----------+
|          1 |          1 |         1 |
|          2 |          2 |         2 |
|          3 |          3 |         3 |
|          4 |          4 |         4 |
|          5 |          5 |         5 |
|          6 |          6 |         6 |
|          7 |          7 |         7 |
|          8 |          8 |         8 |
|          9 |          9 |         9 |
|         10 |         10 |        10 |
|         11 |          1 |        11 |
|         12 |          2 |        12 |
|         13 |          3 |        13 |
|         14 |          4 |        14 |
|         15 |          5 |        15 |
+------------+------------+-----------+
15 rows in set (0.00 sec)
```



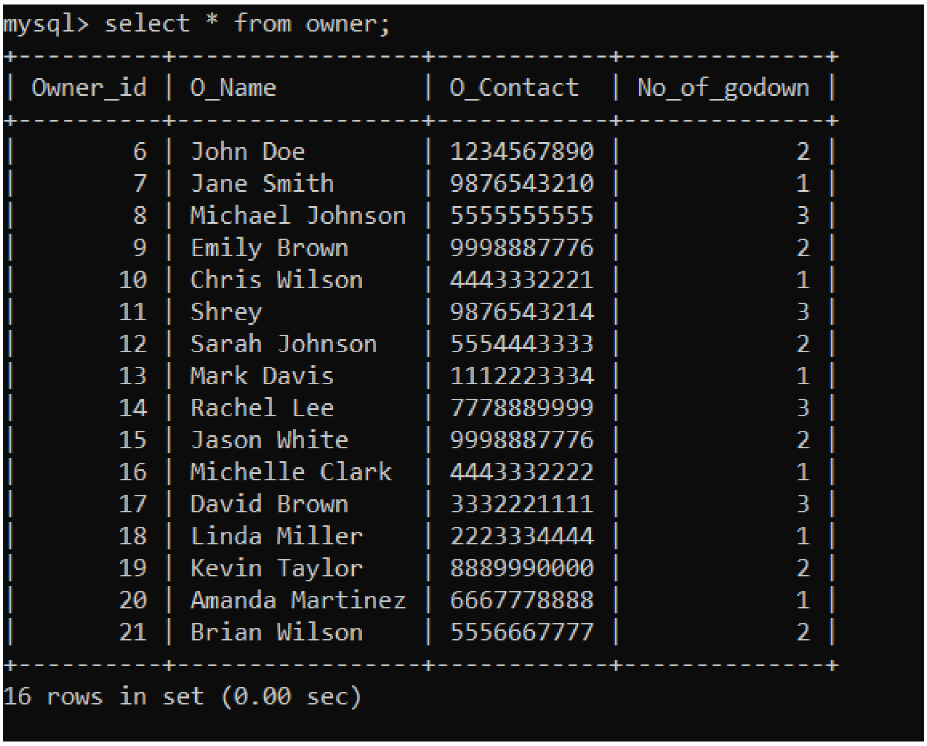
```
mysql> select * from employee;
+-------------+------------------+------------+--------+--------------+----------------+
| Employee_id | E_Name           | E_Contact  | Salary | joining_date | Working_Godown |
+-------------+------------------+------------+--------+--------------+----------------+
|           6 | Alice Johnson    | 1112223333 |  50000 | 2023-01-15   |              1 |
|           7 | Bob Smith        | 4445556666 |  45000 | 2023-02-20   |              2 |
|           8 | Charlie Brown    | 7778889999 |  48000 | 2023-03-25   |              3 |
|           9 | Diana Wilson     | 3332221111 |  52000 | 2023-04-30   |              4 |
|          10 | Eva Garcia       | 6667778888 |  49000 | 2023-05-05   |              5 |
|          11 | Samantha Adams   | 3334445555 |  47000 | 2023-06-10   |              6 |
|          12 | Robert Hernandez | 8889990001 |  51000 | 2023-07-15   |              7 |
|          13 | Cynthia Garcia   | 4445556667 |  49000 | 2023-08-20   |              8 |
|          14 | Matthew Lee      | 1112223335 |  48000 | 2023-09-25   |              9 |
|          15 | Jessica Davis    | 5556667778 |  52000 | 2023-10-30   |             10 |
|          16 | Patrick White    | 7778889990 |  48000 | 2023-11-05   |             11 |
|          17 | Laura Thompson   | 2223334445 |  50000 | 2023-12-10   |             12 |
|          18 | Justin Moore     | 9990001112 |  53000 | 2024-01-15   |             13 |
|          19 | Kelly Hall       | 6667778889 |  49000 | 2024-02-20   |             14 |
|          20 | Brandon Scott    | 3334445556 |  50000 | 2024-03-25   |             15 |
+-------------+------------------+------------+--------+--------------+----------------+
15 rows in set (0.00 sec)
```



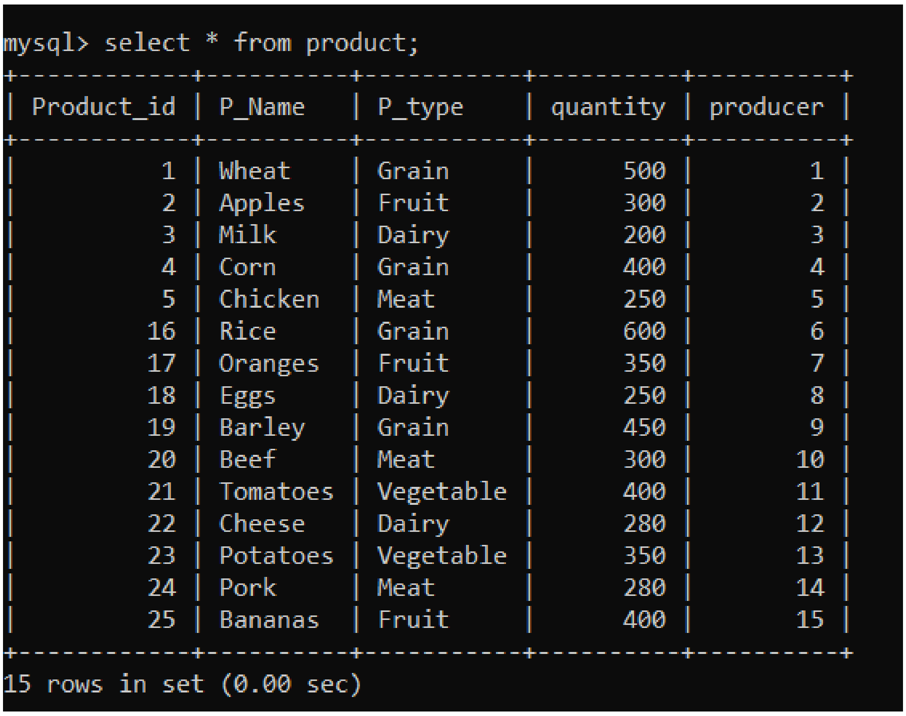
```
mysql> select * from farmer;
+-----------+---------------------+------------+----------------+---------+
| Farmer_id | F_Name              | F_Contact  | F_City         | F_state |
+-----------+---------------------+------------+----------------+---------+
|         6 | John Farmer         | 1234567890 | Springfield    | IL      |
|         7 | Emma Green          | 9876543210 | Seattle        | WA      |
|         8 | Samuel Carter       | 5555555555 | Dallas         | TX      |
|         9 | Olivia Martinez     | 9998887776 | Denver         | CO      |
|        10 | Daniel Taylor       | 4443332221 | Portland       | OR      |
|        11 | Emma Taylor         | 1234567891 | Miami          | FL      |
|        12 | Daniel Brown        | 9876543211 | Chicago        | IL      |
|        13 | Olivia Lee          | 5555555556 | Houston        | TX      |
|        14 | Michael Garcia      | 9998887777 | Los Angeles    | CA      |
|        15 | Sophia Martinez     | 4443332223 | New York       | NY      |
|        16 | James Miller        | 3332221112 | San Francisco  | CA      |
|        17 | Isabella Hernandez  | 2223334446 | Seattle        | WA      |
|        18 | Logan Johnson       | 8889990002 | Dallas         | TX      |
|        19 | Abigail Davis       | 6667778880 | Denver         | CO      |
|        20 | William Clark       | 5556667779 | Phoenix        | AZ      |
+-----------+---------------------+------------+----------------+---------+
15 rows in set (0.00 sec)
```



```
mysql> select * from godown;
+-----------+---------------+---------+----------+--------+----------+
| Godown_id | G_city        | G_state | capacity | G_type | Owner_id |
+-----------+---------------+---------+----------+--------+----------+
|         1 | New York      | NY      |     1000 | Type A |        1 |
|         2 | Los Angeles   | CA      |     1500 | Type B |        2 |
|         3 | Chicago       | IL      |     1200 | Type A |        3 |
|         4 | Houston       | TX      |      800 | Type C |        4 |
|         5 | Miami         | FL      |     1000 | Type B |        5 |
|         6 | San Francisco | CA      |     1200 | Type A |        6 |
|         7 | Seattle       | WA      |     1000 | Type B |        7 |
|         8 | Dallas        | TX      |      800 | Type C |        8 |
|         9 | Denver        | CO      |     1500 | Type B |        9 |
|        10 | Atlanta       | GA      |     1100 | Type A |       10 |
|        11 | Boston        | MA      |     1300 | Type C |       11 |
|        12 | Phoenix       | AZ      |     1000 | Type B |       12 |
|        13 | Las Vegas     | NV      |     1200 | Type A |       13 |
|        14 | Philadelphia  | PA      |      900 | Type C |       14 |
|        15 | San Diego     | CA      |     1100 | Type A |       15 |
+-----------+---------------+---------+----------+--------+----------+
15 rows in set (0.00 sec)
```



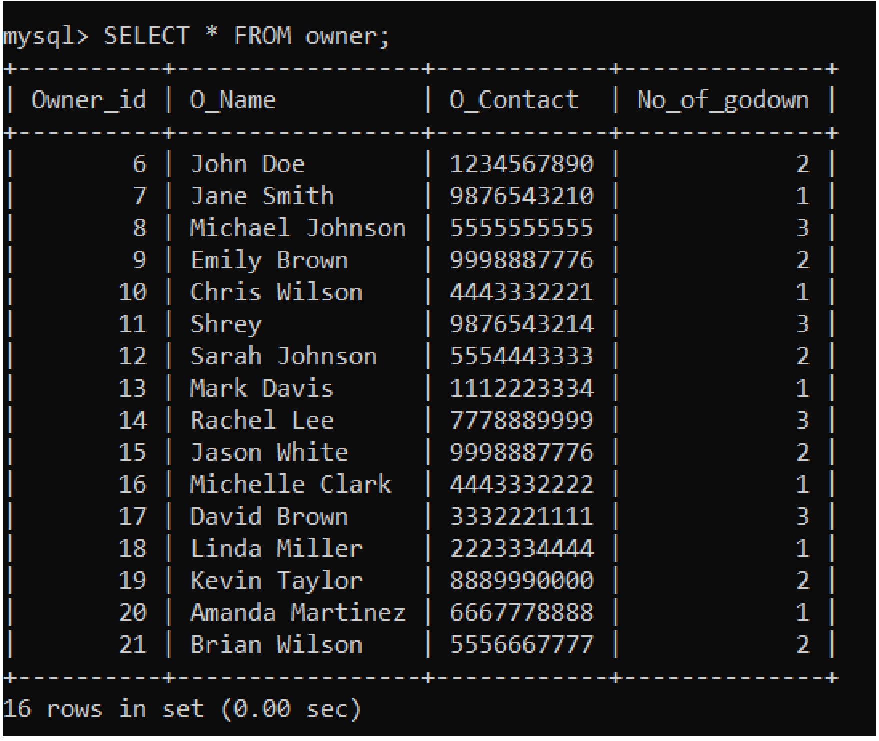
```
mysql> select * from owner;
+----------+----------------+------------+--------------+
| Owner_id | O_Name         | O_Contact  | No_of_godown |
+----------+----------------+------------+--------------+
|        6 | John Doe       | 1234567890 |            2 |
|        7 | Jane Smith     | 9876543210 |            1 |
|        8 | Michael Johnson| 5555555555 |            3 |
|        9 | Emily Brown    | 9998887776 |            2 |
|       10 | Chris Wilson   | 4443332221 |            1 |
|       11 | Shrey          | 9876543214 |            3 |
|       12 | Sarah Johnson  | 5554443333 |            2 |
|       13 | Mark Davis     | 1112223334 |            1 |
|       14 | Rachel Lee     | 7778889999 |            3 |
|       15 | Jason White    | 9998887776 |            2 |
|       16 | Michelle Clark | 4443332222 |            1 |
|       17 | David Brown    | 3332221111 |            3 |
|       18 | Linda Miller   | 2223334444 |            1 |
|       19 | Kevin Taylor   | 8889990000 |            2 |
|       20 | Amanda Martinez| 6667778888 |            1 |
|       21 | Brian Wilson   | 5556667777 |            2 |
+----------+----------------+------------+--------------+
16 rows in set (0.00 sec)
```



```
mysql> select * from product;
+------------+----------+-----------+----------+----------+
| Product_id | P_Name   | P_type    | quantity | producer |
+------------+----------+-----------+----------+----------+
|          1 | Wheat    | Grain     |      500 |        1 |
|          2 | Apples   | Fruit     |      300 |        2 |
|          3 | Milk     | Dairy     |      200 |        3 |
|          4 | Corn     | Grain     |      400 |        4 |
|          5 | Chicken  | Meat      |      250 |        5 |
|         16 | Rice     | Grain     |      600 |        6 |
|         17 | Oranges  | Fruit     |      350 |        7 |
|         18 | Eggs     | Dairy     |      250 |        8 |
|         19 | Barley   | Grain     |      450 |        9 |
|         20 | Beef     | Meat      |      300 |       10 |
|         21 | Tomatoes | Vegetable |      400 |       11 |
|         22 | Cheese   | Dairy     |      280 |       12 |
|         23 | Potatoes | Vegetable |      350 |       13 |
|         24 | Pork     | Meat      |      280 |       14 |
|         25 | Bananas  | Fruit     |      400 |       15 |
+------------+----------+-----------+----------+----------+
15 rows in set (0.00 sec)
```

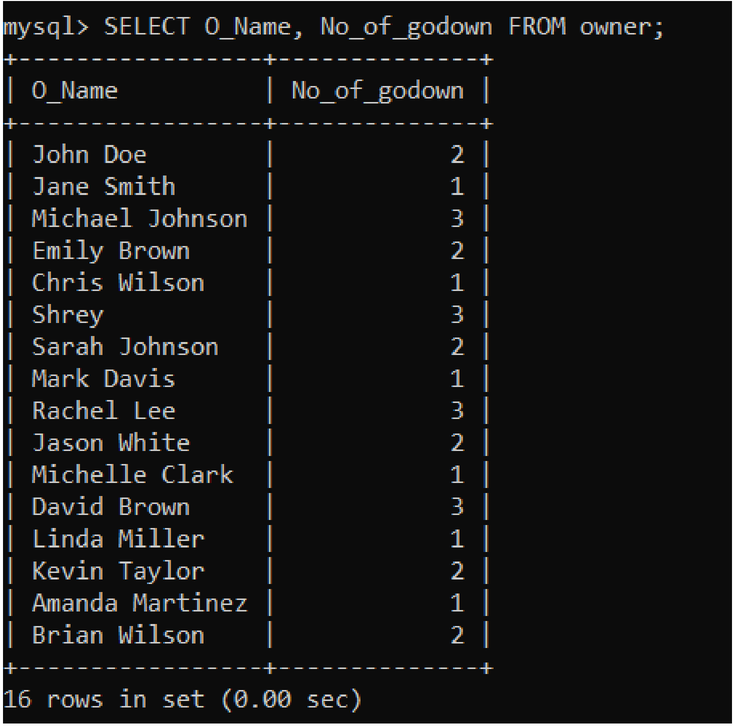
QUERIES

1)SELECT * FROM owner;



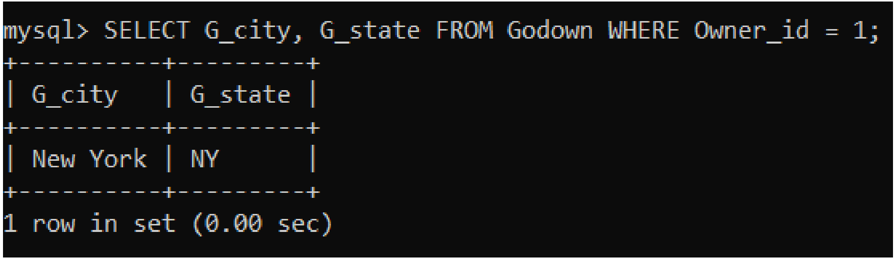
```
mysql> SELECT * FROM owner;
+----------+-----------------+------------+--------------+
| Owner_id | O_Name          | O_Contact  | No_of_godown |
+----------+-----------------+------------+--------------+
|        6 | John Doe        | 1234567890 |            2 |
|        7 | Jane Smith      | 9876543210 |            1 |
|        8 | Michael Johnson | 5555555555 |            3 |
|        9 | Emily Brown     | 9998887776 |            2 |
|       10 | Chris Wilson    | 4443332221 |            1 |
|       11 | Shrey           | 9876543214 |            3 |
|       12 | Sarah Johnson   | 5554443333 |            2 |
|       13 | Mark Davis      | 1112223334 |            1 |
|       14 | Rachel Lee      | 7778889999 |            3 |
|       15 | Jason White     | 9998887776 |            2 |
|       16 | Michelle Clark  | 4443332222 |            1 |
|       17 | David Brown     | 3332221111 |            3 |
|       18 | Linda Miller    | 2223334444 |            1 |
|       19 | Kevin Taylor    | 8889990000 |            2 |
|       20 | Amanda Martinez | 6667778888 |            1 |
|       21 | Brian Wilson    | 5556667777 |            2 |
+----------+-----------------+------------+--------------+
16 rows in set (0.00 sec)
```

2)SELECT O_Name, No_of_godown FROM owner;



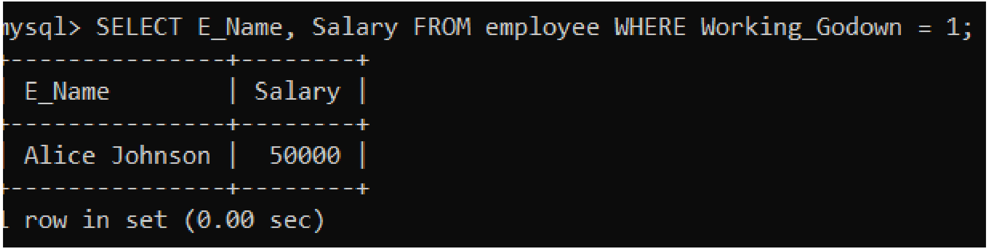
```
mysql> SELECT O_Name, No_of_godown FROM owner;
+-----------------+--------------+
| O_Name          | No_of_godown |
+-----------------+--------------+
| John Doe        |            2 |
| Jane Smith      |            1 |
| Michael Johnson |            3 |
| Emily Brown     |            2 |
| Chris Wilson    |            1 |
| Shrey           |            3 |
| Sarah Johnson   |            2 |
| Mark Davis      |            1 |
| Rachel Lee      |            3 |
| Jason White     |            2 |
| Michelle Clark  |            1 |
| David Brown     |            3 |
| Linda Miller    |            1 |
| Kevin Taylor    |            2 |
| Amanda Martinez |            1 |
| Brian Wilson    |            2 |
+-----------------+--------------+
16 rows in set (0.00 sec)
```

3)SELECT G_city, G_state FROM Godown WHERE Owner_id = 1;



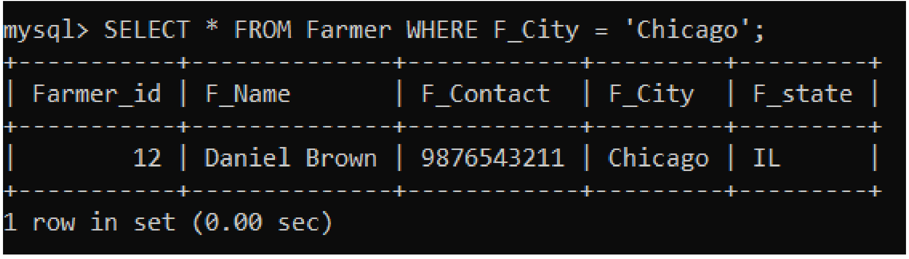
```
mysql> SELECT G_city, G_state FROM Godown WHERE Owner_id = 1;
+----------+---------+
| G_city   | G_state |
+----------+---------+
| New York | NY      |
+----------+---------+
1 row in set (0.00 sec)
```

4)SELECT E_Name, Salary FROM employee WHERE Working_Godown = 1;



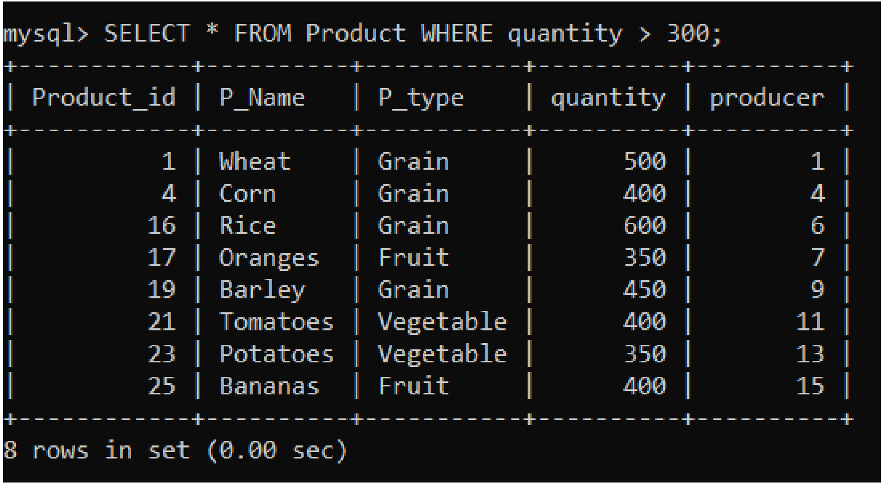
```
ysql> SELECT E_Name, Salary FROM employee WHERE Working_Godown = 1;
---------------+--------+
 E_Name        | Salary |
---------------+--------+
 Alice Johnson |  50000 |
---------------+--------+
 row in set (0.00 sec)
```

5)SELECT * FROM Farmer WHERE F_City = 'Chicago';



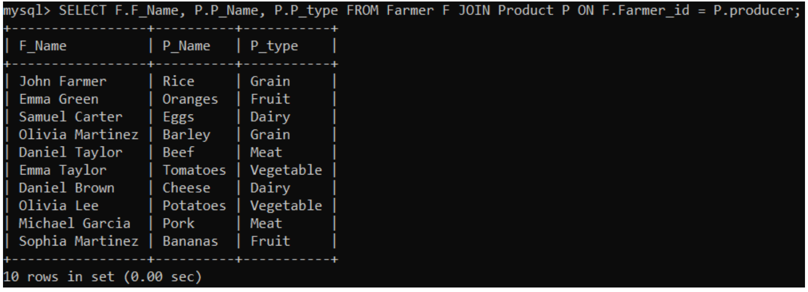
```
mysql> SELECT * FROM Farmer WHERE F_City = 'Chicago';
+-----------+--------------+------------+---------+---------+
| Farmer_id | F_Name       | F_Contact  | F_City  | F_state |
+-----------+--------------+------------+---------+---------+
|        12 | Daniel Brown | 9876543211 | Chicago | IL      |
+-----------+--------------+------------+---------+---------+
1 row in set (0.00 sec)
```

6)SELECT * FROM Product WHERE quantity > 300;



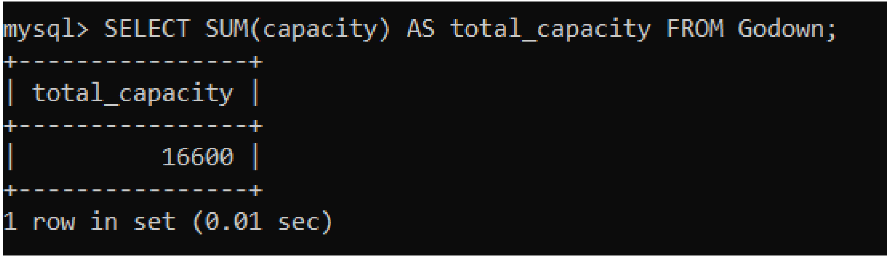
```
mysql> SELECT * FROM Product WHERE quantity > 300;
+------------+----------+-----------+----------+----------+
| Product_id | P_Name   | P_type    | quantity | producer |
+------------+----------+-----------+----------+----------+
|          1 | Wheat    | Grain     |      500 |        1 |
|          4 | Corn     | Grain     |      400 |        4 |
|         16 | Rice     | Grain     |      600 |        6 |
|         17 | Oranges  | Fruit     |      350 |        7 |
|         19 | Barley   | Grain     |      450 |        9 |
|         21 | Tomatoes | Vegetable |      400 |       11 |
|         23 | Potatoes | Vegetable |      350 |       13 |
|         25 | Bananas  | Fruit     |      400 |       15 |
+------------+----------+-----------+----------+----------+
8 rows in set (0.00 sec)
```

7)SELECT F.F_Name, P.P_Name, P.P_type FROM Farmer F JOIN Product P ON F.Farmer_id = P.producer;



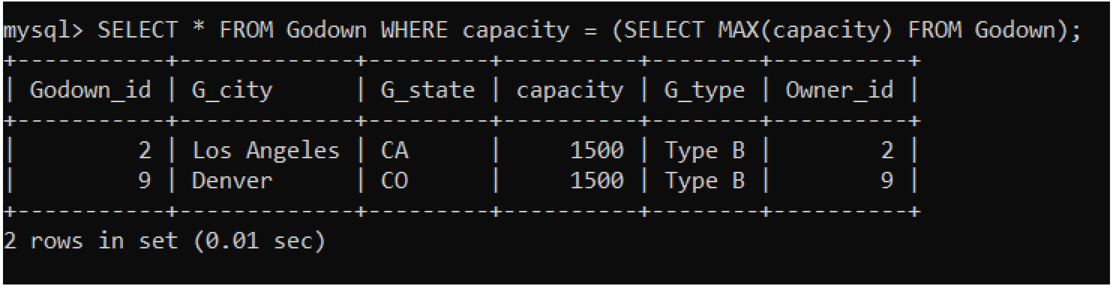
```
mysql> SELECT F.F_Name, P.P_Name, P.P_type FROM Farmer F JOIN Product P ON F.Farmer_id = P.producer;
+-----------------+----------+-----------+
| F_Name          | P_Name   | P_type    |
+-----------------+----------+-----------+
| John Farmer     | Rice     | Grain     |
| Emma Green      | Oranges  | Fruit     |
| Samuel Carter   | Eggs     | Dairy     |
| Olivia Martinez | Barley   | Grain     |
| Daniel Taylor   | Beef     | Meat      |
| Emma Taylor     | Tomatoes | Vegetable |
| Daniel Brown    | Cheese   | Dairy     |
| Olivia Lee      | Potatoes | Vegetable |
| Michael Garcia  | Pork     | Meat      |
| Sophia Martinez | Bananas  | Fruit     |
+-----------------+----------+-----------+
10 rows in set (0.00 sec)
```

8)SELECT SUM(capacity) AS total_capacity FROM Godown;



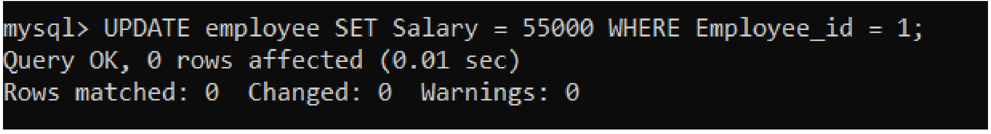
```
mysql> SELECT SUM(capacity) AS total_capacity FROM Godown;
+----------------+
| total_capacity |
+----------------+
|          16600 |
+----------------+
1 row in set (0.01 sec)
```

9)SELECT * FROM Godown WHERE capacity = (SELECT MAX(capacity) FROM Godown);



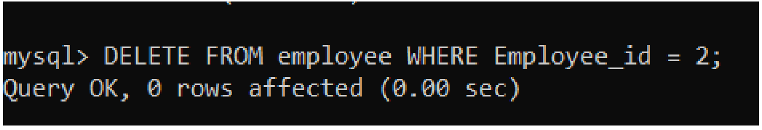
```
mysql> SELECT * FROM Godown WHERE capacity = (SELECT MAX(capacity) FROM Godown);
+-----------+-------------+---------+----------+--------+----------+
| Godown_id | G_city      | G_state | capacity | G_type | Owner_id |
+-----------+-------------+---------+----------+--------+----------+
|         2 | Los Angeles | CA      |     1500 | Type B |        2 |
|         9 | Denver      | CO      |     1500 | Type B |        9 |
+-----------+-------------+---------+----------+--------+----------+
2 rows in set (0.01 sec)
```

10)UPDATE employee SET Salary = 55000 WHERE Employee_id = 1;



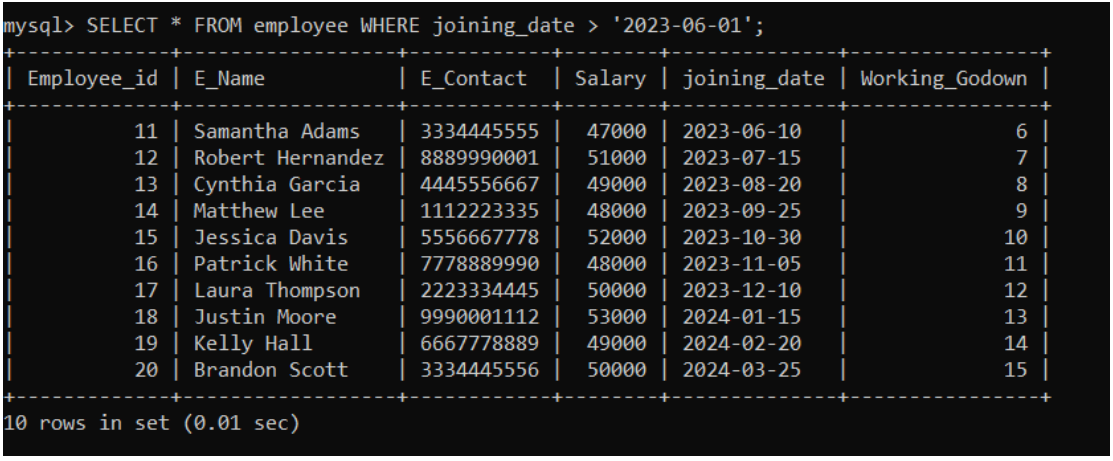
```
mysql> UPDATE employee SET Salary = 55000 WHERE Employee_id = 1;
Query OK, 0 rows affected (0.01 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

11)DELETE FROM employee WHERE Employee_id = 2;



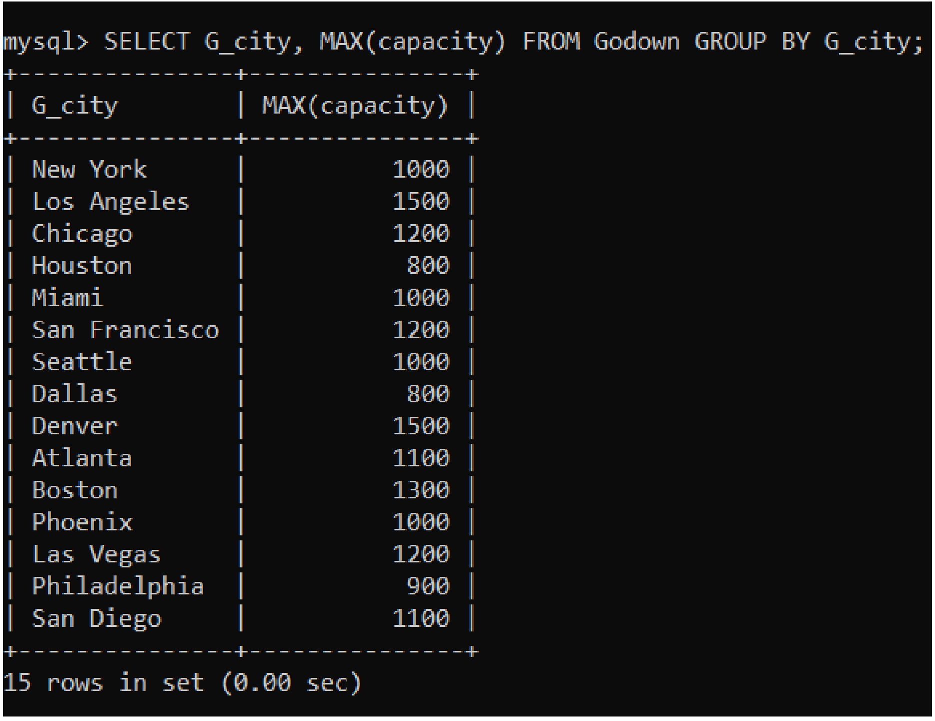
```
mysql> DELETE FROM employee WHERE Employee_id = 2;
Query OK, 0 rows affected (0.00 sec)
```

12)SELECT * FROM employee WHERE joining_date > '2023-06-01';



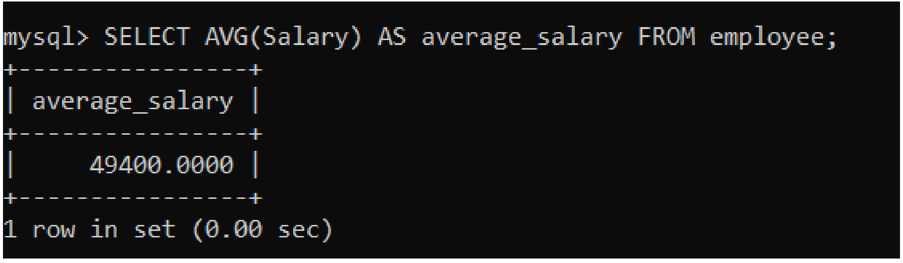
```
mysql> SELECT * FROM employee WHERE joining_date > '2023-06-01';
+-------------+------------------+------------+--------+--------------+----------------+
| Employee_id | E_Name           | E_Contact  | Salary | joining_date | Working_Godown |
+-------------+------------------+------------+--------+--------------+----------------+
|          11 | Samantha Adams   | 3334445555 |  47000 | 2023-06-10   |              6 |
|          12 | Robert Hernandez | 8889990001 |  51000 | 2023-07-15   |              7 |
|          13 | Cynthia Garcia   | 4445556667 |  49000 | 2023-08-20   |              8 |
|          14 | Matthew Lee      | 1112223335 |  48000 | 2023-09-25   |              9 |
|          15 | Jessica Davis    | 5556667778 |  52000 | 2023-10-30   |             10 |
|          16 | Patrick White    | 7778889990 |  48000 | 2023-11-05   |             11 |
|          17 | Laura Thompson   | 2223334445 |  50000 | 2023-12-10   |             12 |
|          18 | Justin Moore     | 9990001112 |  53000 | 2024-01-15   |             13 |
|          19 | Kelly Hall       | 6667778889 |  49000 | 2024-02-20   |             14 |
|          20 | Brandon Scott    | 3334445556 |  50000 | 2024-03-25   |             15 |
+-------------+------------------+------------+--------+--------------+----------------+
10 rows in set (0.01 sec)
```

13)SELECT G_city, MAX(capacity) FROM Godown GROUP BY G_city;



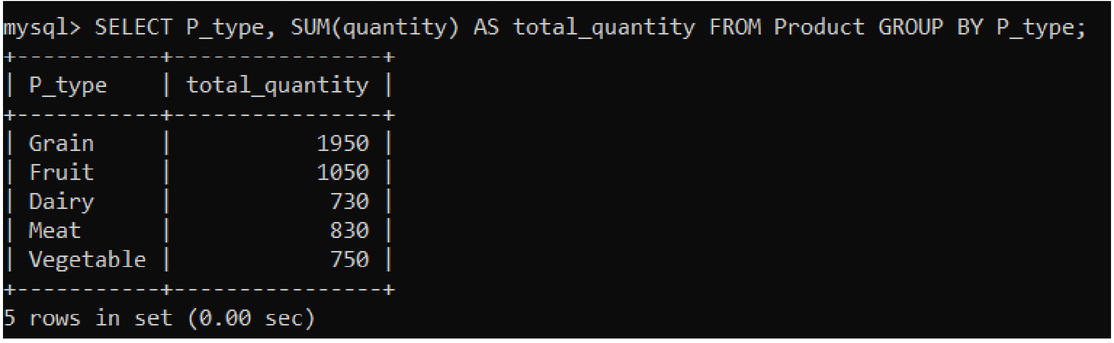
```
mysql> SELECT G_city, MAX(capacity) FROM Godown GROUP BY G_city;
+---------------+---------------+
| G_city        | MAX(capacity) |
+---------------+---------------+
| New York      |          1000 |
| Los Angeles   |          1500 |
| Chicago       |          1200 |
| Houston       |           800 |
| Miami         |          1000 |
| San Francisco |          1200 |
| Seattle       |          1000 |
| Dallas        |           800 |
| Denver        |          1500 |
| Atlanta       |          1100 |
| Boston        |          1300 |
| Phoenix       |          1000 |
| Las Vegas     |          1200 |
| Philadelphia  |           900 |
| San Diego     |          1100 |
+---------------+---------------+
15 rows in set (0.00 sec)
```

14)SELECT AVG(Salary) AS average_salary FROM employee;



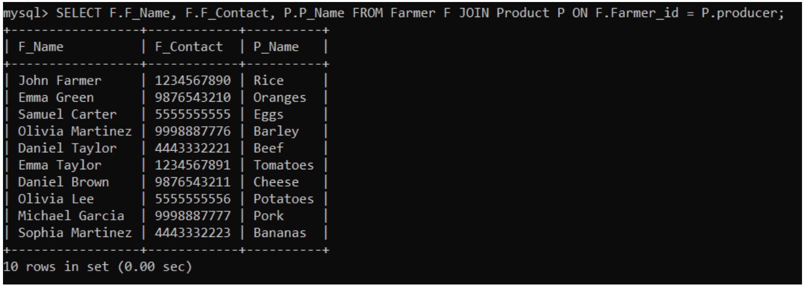
```
mysql> SELECT AVG(Salary) AS average_salary FROM employee;
+----------------+
| average_salary |
+----------------+
|     49400.0000 |
+----------------+
1 row in set (0.00 sec)
```

15)SELECT P_type, SUM(quantity) AS total_quantity FROM Product GROUP BY P_type;



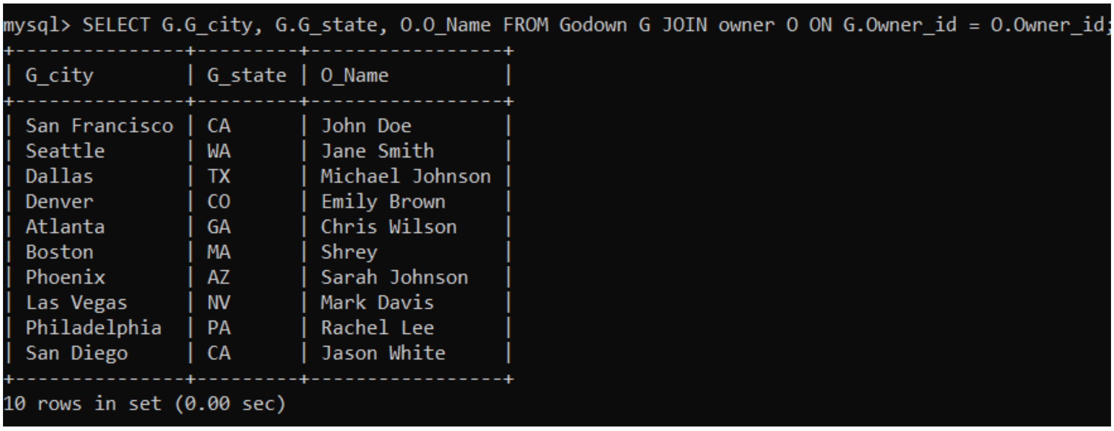
```
mysql> SELECT P_type, SUM(quantity) AS total_quantity FROM Product GROUP BY P_type;
+-----------+----------------+
| P_type    | total_quantity |
+-----------+----------------+
| Grain     |           1950 |
| Fruit     |           1050 |
| Dairy     |            730 |
| Meat      |            830 |
| Vegetable |            750 |
+-----------+----------------+
5 rows in set (0.00 sec)
```

16)SELECT F.F_Name, F.F_Contact, P.P_Name FROM Farmer F JOIN Product P ON F.Farmer_id = P.producer;



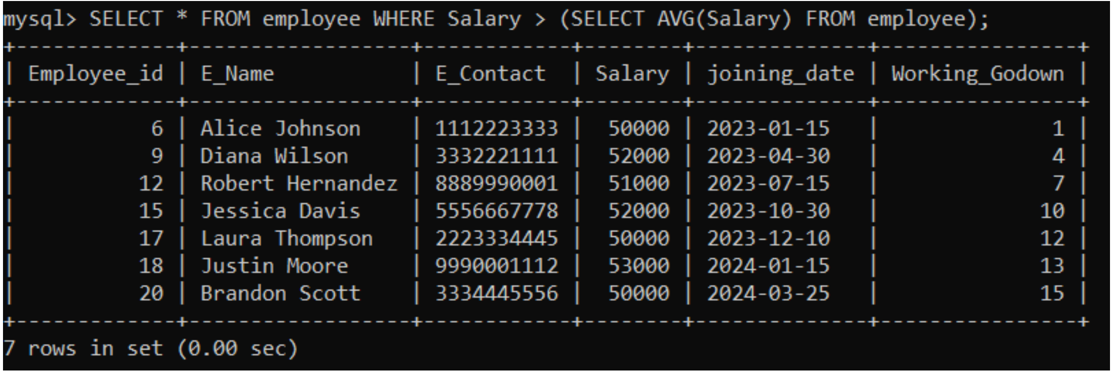
```
mysql> SELECT F.F_Name, F.F_Contact, P.P_Name FROM Farmer F JOIN Product P ON F.Farmer_id = P.producer;
+-----------------+------------+----------+
| F_Name          | F_Contact  | P_Name   |
+-----------------+------------+----------+
| John Farmer     | 1234567890 | Rice     |
| Emma Green      | 9876543210 | Oranges  |
| Samuel Carter   | 5555555555 | Eggs     |
| Olivia Martinez | 9998887776 | Barley   |
| Daniel Taylor   | 4443332221 | Beef     |
| Emma Taylor     | 1234567891 | Tomatoes |
| Daniel Brown    | 9876543211 | Cheese   |
| Olivia Lee      | 5555555556 | Potatoes |
| Michael Garcia  | 9998887777 | Pork     |
| Sophia Martinez | 4443332223 | Bananas  |
+-----------------+------------+----------+
10 rows in set (0.00 sec)
```

17)SELECT G.G_city, G.G_state, O.O_Name FROM Godown G JOIN owner O ON G.Owner_id = O.Owner_id;



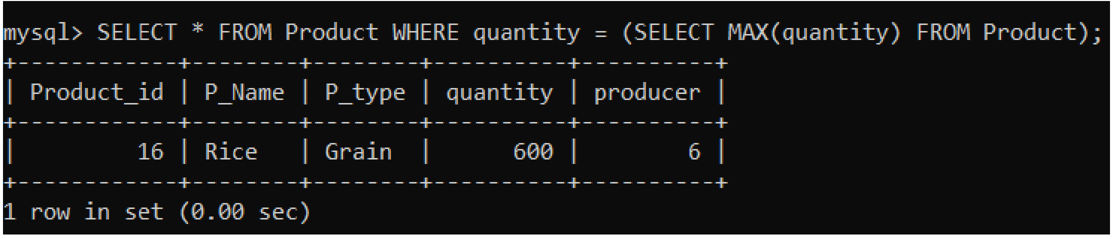
```
mysql> SELECT G.G_city, G.G_state, O.O_Name FROM Godown G JOIN owner O ON G.Owner_id = O.Owner_id;
+---------------+---------+-----------------+
| G_city        | G_state | O_Name          |
+---------------+---------+-----------------+
| San Francisco | CA      | John Doe        |
| Seattle       | WA      | Jane Smith      |
| Dallas        | TX      | Michael Johnson |
| Denver        | CO      | Emily Brown     |
| Atlanta       | GA      | Chris Wilson    |
| Boston        | MA      | Shrey           |
| Phoenix       | AZ      | Sarah Johnson   |
| Las Vegas     | NV      | Mark Davis      |
| Philadelphia  | PA      | Rachel Lee      |
| San Diego     | CA      | Jason White     |
+---------------+---------+-----------------+
10 rows in set (0.00 sec)
```

18)SELECT * FROM employee WHERE Salary > (SELECT AVG(Salary) FROM employee);



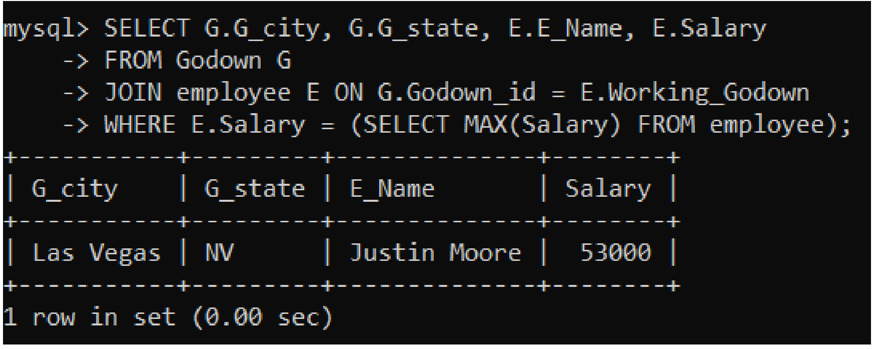
```
mysql> SELECT * FROM employee WHERE Salary > (SELECT AVG(Salary) FROM employee);
+-------------+------------------+------------+--------+--------------+-----------------+
| Employee_id | E_Name           | E_Contact  | Salary | joining_date | Working_Godown  |
+-------------+------------------+------------+--------+--------------+-----------------+
|           6 | Alice Johnson    | 1112223333 |  50000 | 2023-01-15   |               1 |
|           9 | Diana Wilson     | 3332221111 |  52000 | 2023-04-30   |               4 |
|          12 | Robert Hernandez | 8889990001 |  51000 | 2023-07-15   |               7 |
|          15 | Jessica Davis    | 5556667778 |  52000 | 2023-10-30   |              10 |
|          17 | Laura Thompson   | 2223334445 |  50000 | 2023-12-10   |              12 |
|          18 | Justin Moore     | 9990001112 |  53000 | 2024-01-15   |              13 |
|          20 | Brandon Scott    | 3334445556 |  50000 | 2024-03-25   |              15 |
+-------------+------------------+------------+--------+--------------+-----------------+
7 rows in set (0.00 sec)
```

19)SELECT * FROM Product WHERE quantity = (SELECT MAX(quantity) FROM Product);



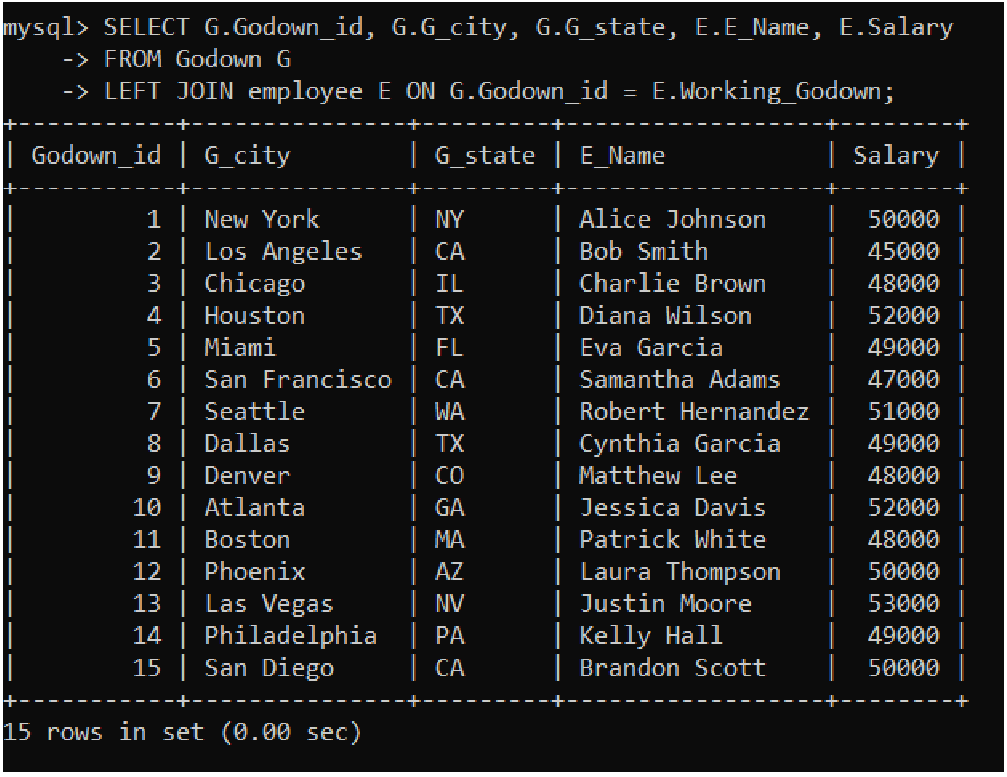
```
mysql> SELECT * FROM Product WHERE quantity = (SELECT MAX(quantity) FROM Product);
+------------+--------+--------+----------+----------+
| Product_id | P_Name | P_type | quantity | producer |
+------------+--------+--------+----------+----------+
|         16 | Rice   | Grain  |      600 |        6 |
+------------+--------+--------+----------+----------+
1 row in set (0.00 sec)
```

20)SELECT G.G_city, G.G_state, E.E_Name, E.Salary
FROM Godown G
JOIN employee E ON G.Godown_id = E.Working_Godown
WHERE E.Salary = (SELECT MAX(Salary) FROM employee);



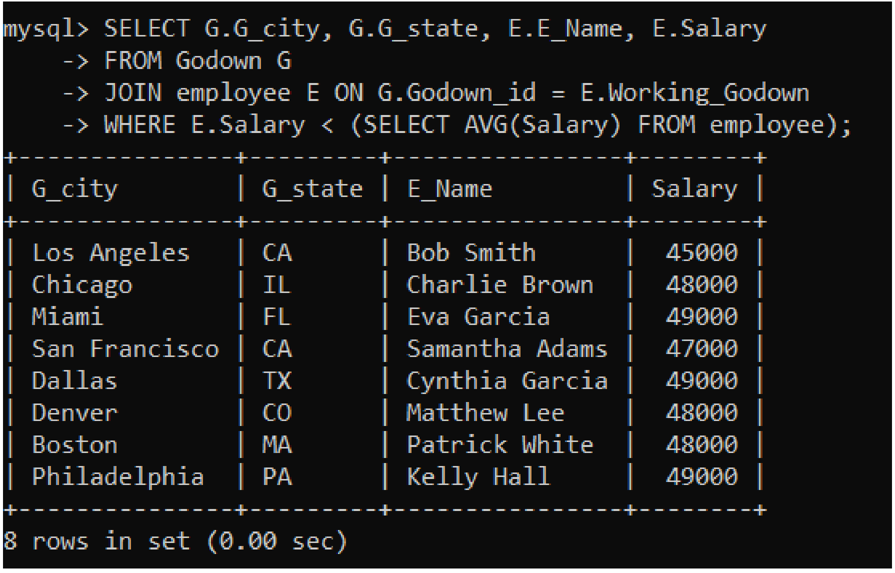
```
mysql> SELECT G.G_city, G.G_state, E.E_Name, E.Salary
    -> FROM Godown G
    -> JOIN employee E ON G.Godown_id = E.Working_Godown
    -> WHERE E.Salary = (SELECT MAX(Salary) FROM employee);
+-----------+---------+--------------+--------+
| G_city    | G_state | E_Name       | Salary |
+-----------+---------+--------------+--------+
| Las Vegas | NV      | Justin Moore |  53000 |
+-----------+---------+--------------+--------+
1 row in set (0.00 sec)
```

21)SELECT G.Godown_id, G.G_city, G.G_state, E.E_Name, E.Salary
FROM Godown G
LEFT JOIN employee E ON G.Godown_id = E.Working_Godown;



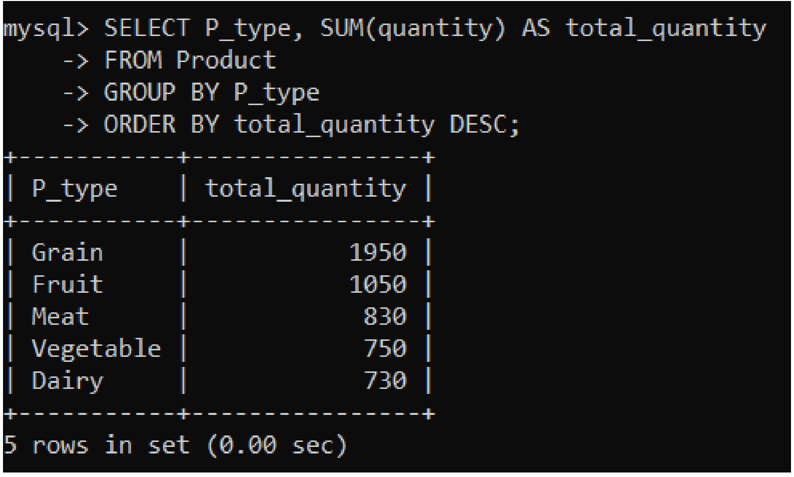
```
mysql> SELECT G.Godown_id, G.G_city, G.G_state, E.E_Name, E.Salary
    -> FROM Godown G
    -> LEFT JOIN employee E ON G.Godown_id = E.Working_Godown;
+-----------+---------------+---------+------------------+--------+
| Godown_id | G_city        | G_state | E_Name           | Salary |
+-----------+---------------+---------+------------------+--------+
|         1 | New York      | NY      | Alice Johnson    |  50000 |
|         2 | Los Angeles   | CA      | Bob Smith        |  45000 |
|         3 | Chicago       | IL      | Charlie Brown    |  48000 |
|         4 | Houston       | TX      | Diana Wilson     |  52000 |
|         5 | Miami         | FL      | Eva Garcia       |  49000 |
|         6 | San Francisco | CA      | Samantha Adams   |  47000 |
|         7 | Seattle       | WA      | Robert Hernandez |  51000 |
|         8 | Dallas        | TX      | Cynthia Garcia   |  49000 |
|         9 | Denver        | CO      | Matthew Lee      |  48000 |
|        10 | Atlanta       | GA      | Jessica Davis    |  52000 |
|        11 | Boston        | MA      | Patrick White    |  48000 |
|        12 | Phoenix       | AZ      | Laura Thompson   |  50000 |
|        13 | Las Vegas     | NV      | Justin Moore     |  53000 |
|        14 | Philadelphia  | PA      | Kelly Hall       |  49000 |
|        15 | San Diego     | CA      | Brandon Scott    |  50000 |
+-----------+---------------+---------+------------------+--------+
15 rows in set (0.00 sec)
```

22)SELECT G.G_city, G.G_state, E.E_Name, E.Salary
FROM Godown G
JOIN employee E ON G.Godown_id = E.Working_Godown
WHERE E.Salary < (SELECT AVG(Salary) FROM employee);



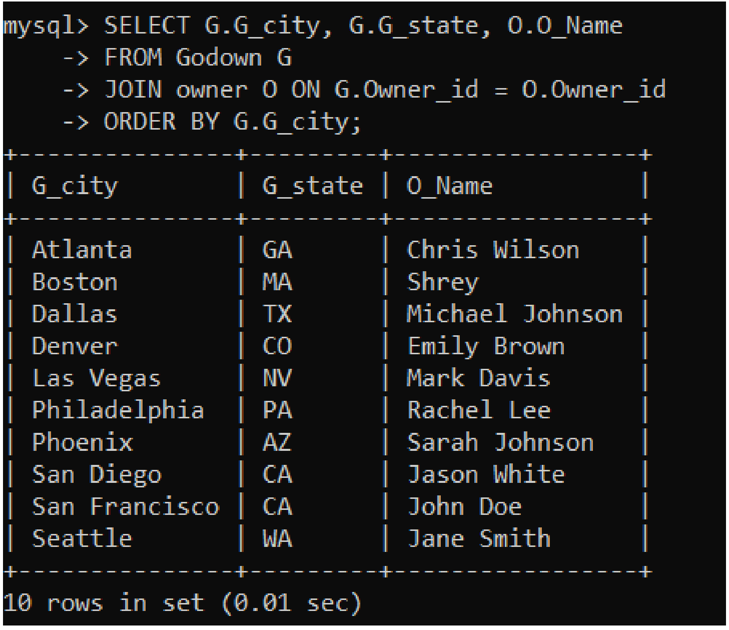
```
mysql> SELECT G.G_city, G.G_state, E.E_Name, E.Salary
    -> FROM Godown G
    -> JOIN employee E ON G.Godown_id = E.Working_Godown
    -> WHERE E.Salary < (SELECT AVG(Salary) FROM employee);
+---------------+---------+----------------+--------+
| G_city        | G_state | E_Name         | Salary |
+---------------+---------+----------------+--------+
| Los Angeles   | CA      | Bob Smith      |  45000 |
| Chicago       | IL      | Charlie Brown  |  48000 |
| Miami         | FL      | Eva Garcia     |  49000 |
| San Francisco | CA      | Samantha Adams |  47000 |
| Dallas        | TX      | Cynthia Garcia |  49000 |
| Denver        | CO      | Matthew Lee    |  48000 |
| Boston        | MA      | Patrick White  |  48000 |
| Philadelphia  | PA      | Kelly Hall     |  49000 |
+---------------+---------+----------------+--------+
8 rows in set (0.00 sec)
```

```
23)SELECT P_type, SUM(quantity) AS total_quantity
FROM Product
GROUP BY P_type
ORDER BY total_quantity DESC;
```



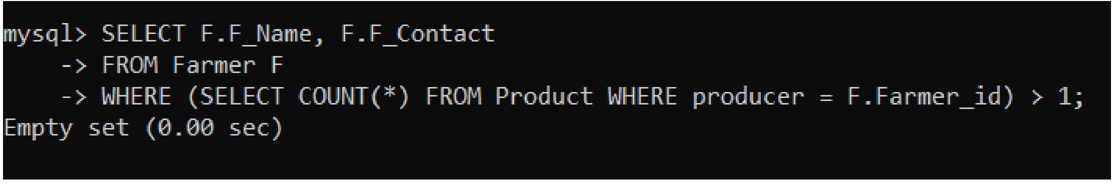
```
mysql> SELECT P_type, SUM(quantity) AS total_quantity
    -> FROM Product
    -> GROUP BY P_type
    -> ORDER BY total_quantity DESC;
+-----------+----------------+
| P_type    | total_quantity |
+-----------+----------------+
| Grain     |           1950 |
| Fruit     |           1050 |
| Meat      |            830 |
| Vegetable |            750 |
| Dairy     |            730 |
+-----------+----------------+
5 rows in set (0.00 sec)
```

```
24)SELECT G.G_city, G.G_state, O.O_Name
FROM Godown G
JOIN owner O ON G.Owner_id = O.Owner_id
ORDER BY G.G_city;
```



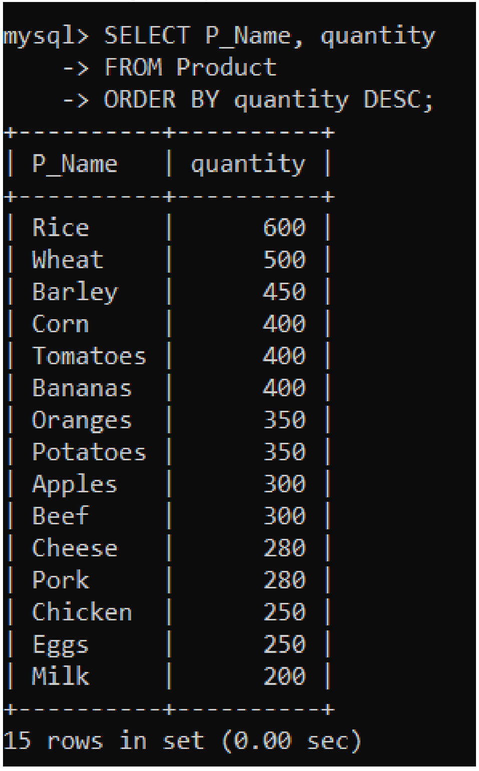
```
mysql> SELECT G.G_city, G.G_state, O.O_Name
    -> FROM Godown G
    -> JOIN owner O ON G.Owner_id = O.Owner_id
    -> ORDER BY G.G_city;
+---------------+---------+-----------------+
| G_city        | G_state | O_Name          |
+---------------+---------+-----------------+
| Atlanta       | GA      | Chris Wilson    |
| Boston        | MA      | Shrey           |
| Dallas        | TX      | Michael Johnson |
| Denver        | CO      | Emily Brown     |
| Las Vegas     | NV      | Mark Davis      |
| Philadelphia  | PA      | Rachel Lee      |
| Phoenix       | AZ      | Sarah Johnson   |
| San Diego     | CA      | Jason White     |
| San Francisco | CA      | John Doe        |
| Seattle       | WA      | Jane Smith      |
+---------------+---------+-----------------+
10 rows in set (0.01 sec)
```

```
25)SELECT F.F_Name, F.F_Contact
FROM Farmer F
WHERE (SELECT COUNT(*) FROM Product WHERE producer = F.Farmer_id) > 1;
```



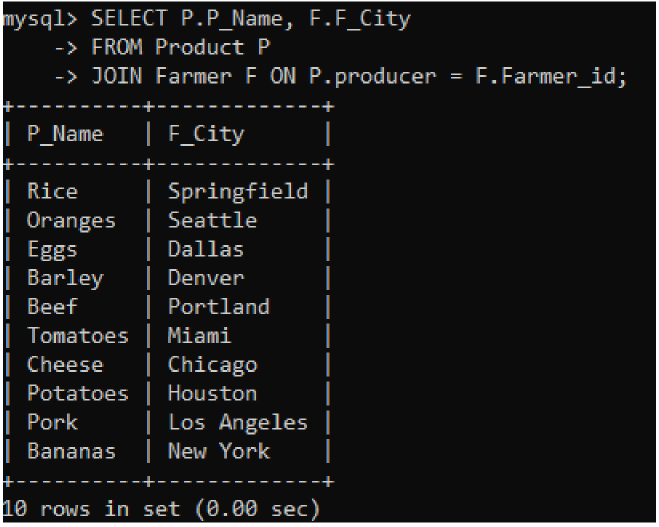
```
mysql> SELECT F.F_Name, F.F_Contact
    -> FROM Farmer F
    -> WHERE (SELECT COUNT(*) FROM Product WHERE producer = F.Farmer_id) > 1;
Empty set (0.00 sec)
```

26)SELECT P_Name, quantity
FROM Product
ORDER BY quantity DESC;



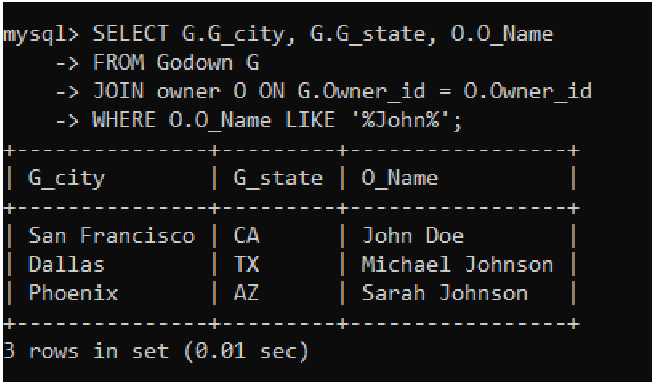
```
mysql> SELECT P_Name, quantity
    -> FROM Product
    -> ORDER BY quantity DESC;
+----------+----------+
| P_Name   | quantity |
+----------+----------+
| Rice     |      600 |
| Wheat    |      500 |
| Barley   |      450 |
| Corn     |      400 |
| Tomatoes |      400 |
| Bananas  |      400 |
| Oranges  |      350 |
| Potatoes |      350 |
| Apples   |      300 |
| Beef     |      300 |
| Cheese   |      280 |
| Pork     |      280 |
| Chicken  |      250 |
| Eggs     |      250 |
| Milk     |      200 |
+----------+----------+
15 rows in set (0.00 sec)
```

27)SELECT P.P_Name, F.F_City
FROM Product P
JOIN Farmer F ON P.producer = F.Farmer_id;



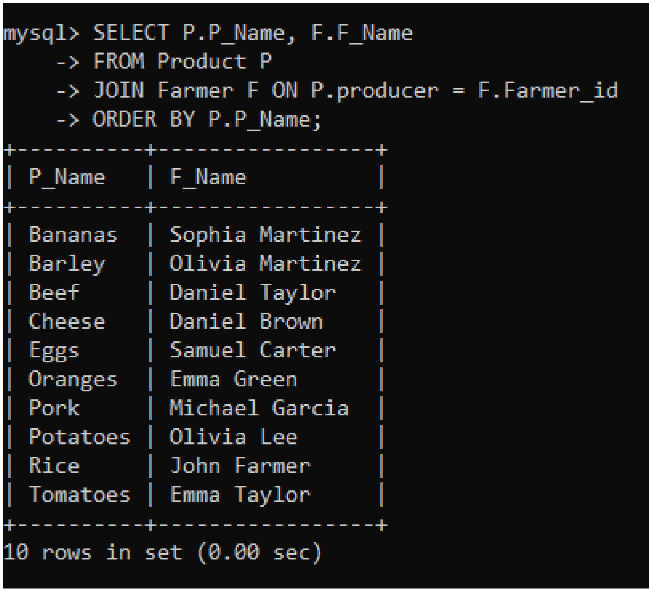
```
mysql> SELECT P.P_Name, F.F_City
    -> FROM Product P
    -> JOIN Farmer F ON P.producer = F.Farmer_id;
+----------+-------------+
| P_Name   | F_City      |
+----------+-------------+
| Rice     | Springfield |
| Oranges  | Seattle     |
| Eggs     | Dallas      |
| Barley   | Denver      |
| Beef     | Portland    |
| Tomatoes | Miami       |
| Cheese   | Chicago     |
| Potatoes | Houston     |
| Pork     | Los Angeles |
| Bananas  | New York    |
+----------+-------------+
10 rows in set (0.00 sec)
```

28)SELECT G.G_city, G.G_state, O.O_Name
FROM Godown G
JOIN owner O ON G.Owner_id = O.Owner_id
WHERE O.O_Name LIKE '%John%';



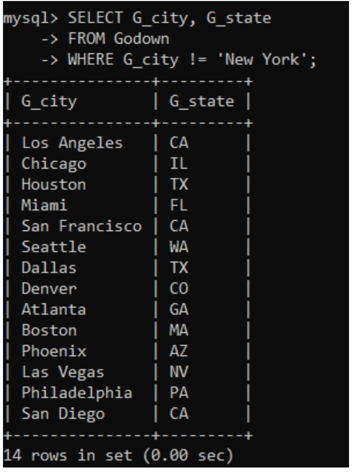
```
mysql> SELECT G.G_city, G.G_state, O.O_Name
    -> FROM Godown G
    -> JOIN owner O ON G.Owner_id = O.Owner_id
    -> WHERE O.O_Name LIKE '%John%';
+---------------+---------+-----------------+
| G_city        | G_state | O_Name          |
+---------------+---------+-----------------+
| San Francisco | CA      | John Doe        |
| Dallas        | TX      | Michael Johnson |
| Phoenix       | AZ      | Sarah Johnson   |
+---------------+---------+-----------------+
3 rows in set (0.01 sec)
```

29)SELECT P.P_Name, F.F_Name
FROM Product P
JOIN Farmer F ON P.producer = F.Farmer_id
ORDER BY P.P_Name;



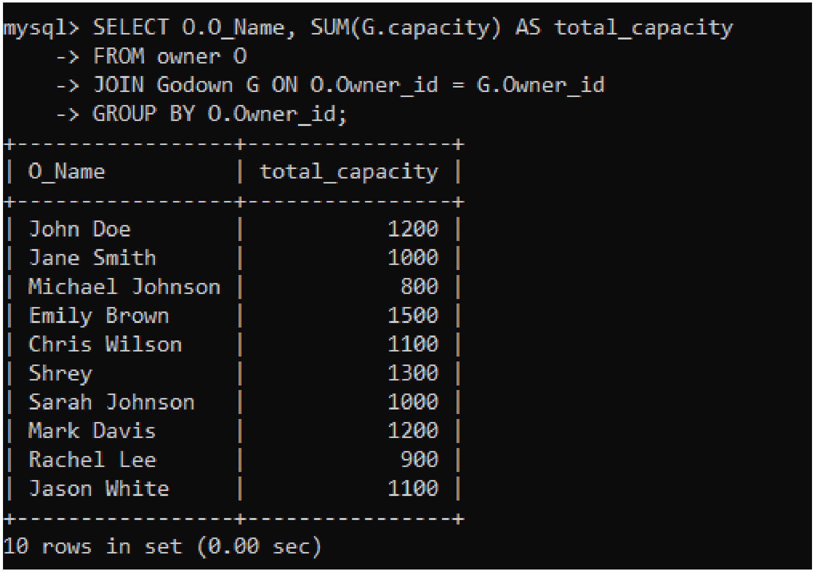
```
mysql> SELECT P.P_Name, F.F_Name
    -> FROM Product P
    -> JOIN Farmer F ON P.producer = F.Farmer_id
    -> ORDER BY P.P_Name;
+----------+-----------------+
| P_Name   | F_Name          |
+----------+-----------------+
| Bananas  | Sophia Martinez |
| Barley   | Olivia Martinez |
| Beef     | Daniel Taylor   |
| Cheese   | Daniel Brown    |
| Eggs     | Samuel Carter   |
| Oranges  | Emma Green      |
| Pork     | Michael Garcia  |
| Potatoes | Olivia Lee      |
| Rice     | John Farmer     |
| Tomatoes | Emma Taylor     |
+----------+-----------------+
10 rows in set (0.00 sec)
```

30)SELECT G_city, G_state
FROM Godown
WHERE G_city != 'New York';



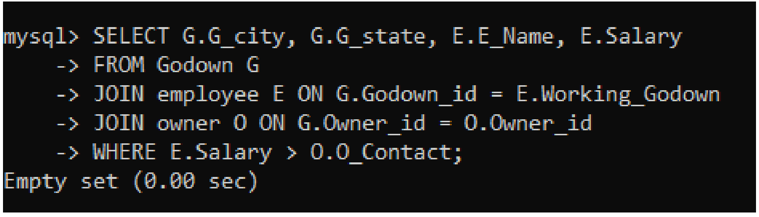
```
mysql> SELECT G_city, G_state
    -> FROM Godown
    -> WHERE G_city != 'New York';
+---------------+---------+
| G_city        | G_state |
+---------------+---------+
| Los Angeles   | CA      |
| Chicago       | IL      |
| Houston       | TX      |
| Miami         | FL      |
| San Francisco | CA      |
| Seattle       | WA      |
| Dallas        | TX      |
| Denver        | CO      |
| Atlanta       | GA      |
| Boston        | MA      |
| Phoenix       | AZ      |
| Las Vegas     | NV      |
| Philadelphia  | PA      |
| San Diego     | CA      |
+---------------+---------+
14 rows in set (0.00 sec)
```

31)SELECT O.O_Name, SUM(G.capacity) AS total_capacity
FROM owner O
JOIN Godown G ON O.Owner_id = G.Owner_id
GROUP BY O.Owner_id;



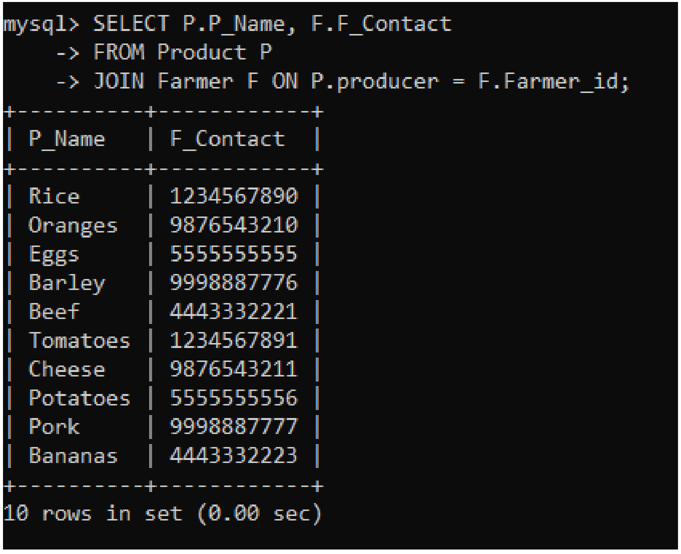
```
mysql> SELECT O.O_Name, SUM(G.capacity) AS total_capacity
    -> FROM owner O
    -> JOIN Godown G ON O.Owner_id = G.Owner_id
    -> GROUP BY O.Owner_id;
+-----------------+----------------+
| O_Name          | total_capacity |
+-----------------+----------------+
| John Doe        |           1200 |
| Jane Smith      |           1000 |
| Michael Johnson |            800 |
| Emily Brown     |           1500 |
| Chris Wilson    |           1100 |
| Shrey           |           1300 |
| Sarah Johnson   |           1000 |
| Mark Davis      |           1200 |
| Rachel Lee      |            900 |
| Jason White     |           1100 |
+-----------------+----------------+
10 rows in set (0.00 sec)
```

32)SELECT G.G_city, G.G_state, E.E_Name, E.Salary
FROM Godown G
JOIN employee E ON G.Godown_id = E.Working_Godown
JOIN owner O ON G.Owner_id = O.Owner_id
WHERE E.Salary > O.O_Contact;



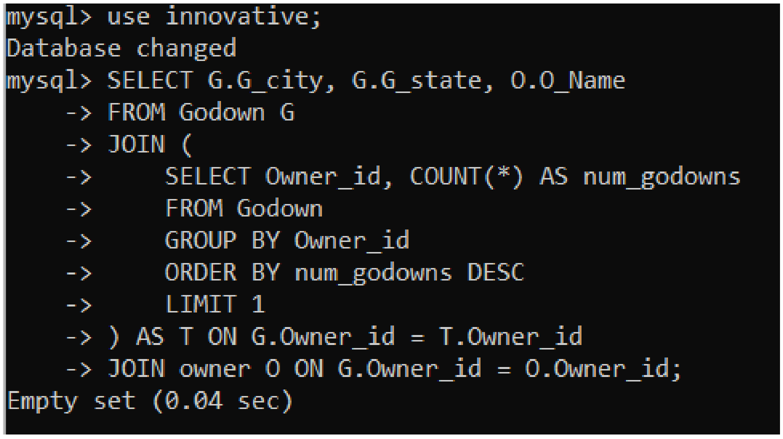
```
mysql> SELECT G.G_city, G.G_state, E.E_Name, E.Salary
    -> FROM Godown G
    -> JOIN employee E ON G.Godown_id = E.Working_Godown
    -> JOIN owner O ON G.Owner_id = O.Owner_id
    -> WHERE E.Salary > O.O_Contact;
Empty set (0.00 sec)
```

33)SELECT P.P_Name, F.F_Contact
FROM Product P
JOIN Farmer F ON P.producer = F.Farmer_id;



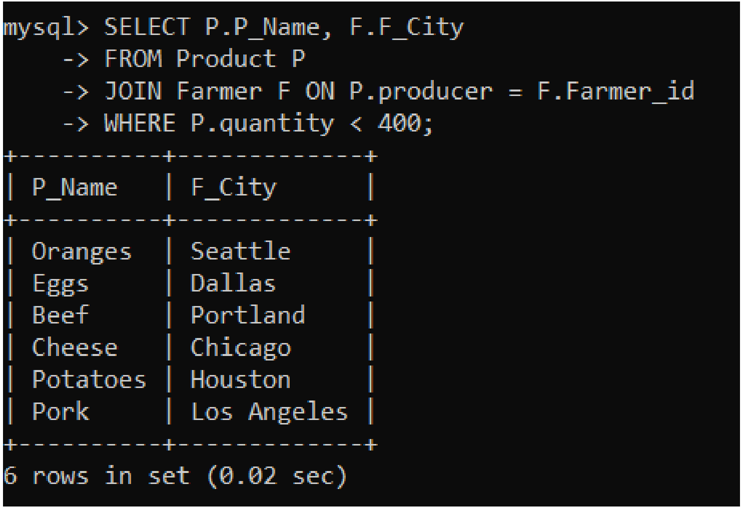
```
mysql> SELECT P.P_Name, F.F_Contact
    -> FROM Product P
    -> JOIN Farmer F ON P.producer = F.Farmer_id;
+----------+------------+
| P_Name   | F_Contact  |
+----------+------------+
| Rice     | 1234567890 |
| Oranges  | 9876543210 |
| Eggs     | 5555555555 |
| Barley   | 9998887776 |
| Beef     | 4443332221 |
| Tomatoes | 1234567891 |
| Cheese   | 9876543211 |
| Potatoes | 5555555556 |
| Pork     | 9998887777 |
| Bananas  | 4443332223 |
+----------+------------+
10 rows in set (0.00 sec)
```

34)SELECT G.G_city, G.G_state, O.O_Name
FROM Godown G
JOIN (
SELECT Owner_id, COUNT(*) AS num_godowns
FROM Godown
GROUP BY Owner_id
ORDER BY num_godowns DESC
LIMIT 1
) AS T ON G.Owner_id = T.Owner_id
JOIN owner O ON G.Owner_id = O.Owner_id;



```
mysql> use innovative;
Database changed
mysql> SELECT G.G_city, G.G_state, O.O_Name
    -> FROM Godown G
    -> JOIN (
    ->     SELECT Owner_id, COUNT(*) AS num_godowns
    ->     FROM Godown
    ->     GROUP BY Owner_id
    ->     ORDER BY num_godowns DESC
    ->     LIMIT 1
    -> ) AS T ON G.Owner_id = T.Owner_id
    -> JOIN owner O ON G.Owner_id = O.Owner_id;
Empty set (0.04 sec)
```

35)SELECT P.P_Name, F.F_City
FROM Product P
JOIN Farmer F ON P.producer = F.Farmer_id
WHERE P.quantity < 400;



```
mysql> SELECT P.P_Name, F.F_City
    -> FROM Product P
    -> JOIN Farmer F ON P.producer = F.Farmer_id
    -> WHERE P.quantity < 400;
+----------+-------------+
| P_Name   | F_City      |
+----------+-------------+
| Oranges  | Seattle     |
| Eggs     | Dallas      |
| Beef     | Portland    |
| Cheese   | Chicago     |
| Potatoes | Houston     |
| Pork     | Los Angeles |
+----------+-------------+
6 rows in set (0.02 sec)
```

Python Code for Integrity Management

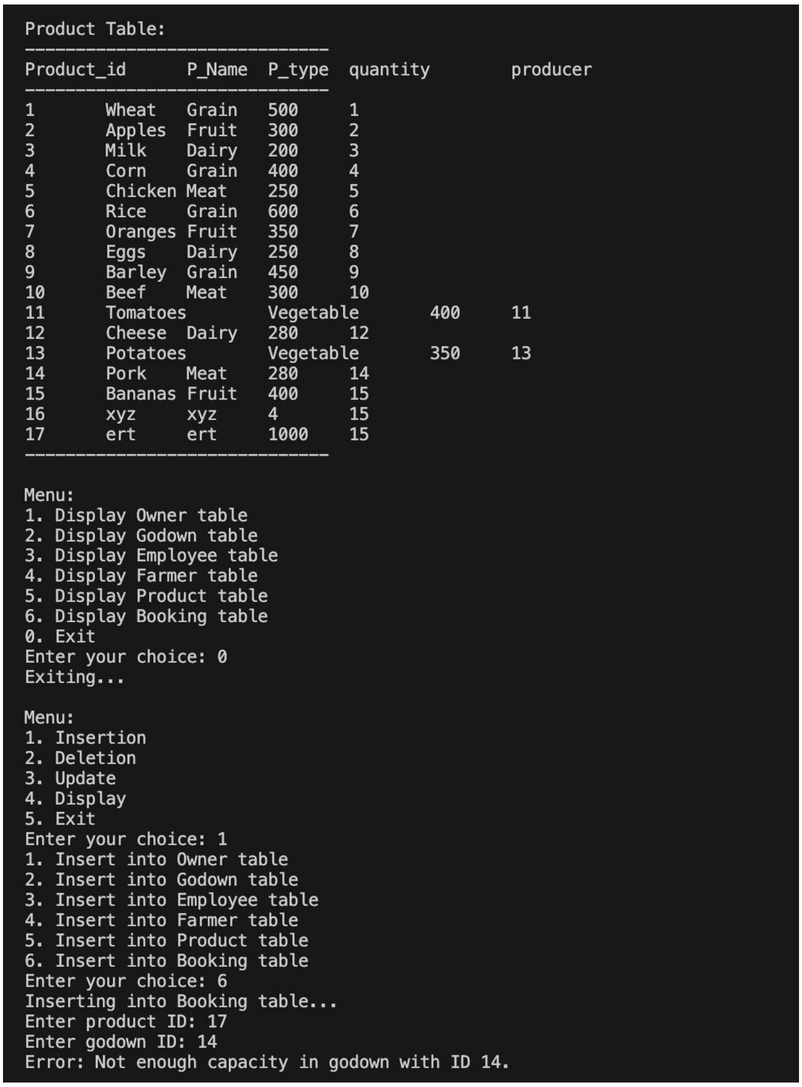
```
import mysql.connector

# Function to establish connection to the database
def connect_to_database():
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="password",
            database="innovative"
        )
        return conn
    except mysql.connector.Error as err:
        print("Error:", err)

# Function for user login
def login():
    # Hardcoded login credentials for demonstration purposes
    # In a real-world scenario, you'd fetch these from the database
    hardcoded_username = "admin"
    hardcoded_password = "password"

    username = input("Enter username: ")
    password = input("Enter password: ")

    if username == hardcoded_username and password == hardcoded_password:
        print("Login successful!")
        return True
    else:
        print("Invalid username or password. Please try again.")
        return False

# Function for insertion
def insertion_menu(conn):
    cursor = conn.cursor()

    print("1. Insert into Owner table")
    print("2. Insert into Godown table")
    print("3. Insert into Employee table")
    print("4. Insert into Farmer table")
    print("5. Insert into Product table")
    print("6. Insert into Booking table")

    choice = input("Enter your choice: ")

    if choice == "1":
        owner_insertion(conn, cursor)
    elif choice == "2":
        godown_insertion(conn, cursor)
    elif choice == "3":
        employee_insertion(conn, cursor)
    elif choice == "4":
        farmer_insertion(conn, cursor)
    elif choice == "5":
        product_insertion(conn, cursor)
    elif choice == "6":
        booking_insertion(conn, cursor)
    else:
        print("Invalid choice.")

    conn.commit()
    cursor.close()

# Insertion function for Owner table
def owner_insertion(conn, cursor):
    print("Inserting into Owner table...")
    o_name = input("Enter owner name: ")
    o_contact = input("Enter owner contact: ")
    no_of_godown = int(input("Enter number of godowns owned: "))

    # Insert into Owner table
    try:
        cursor.execute("INSERT INTO owner (O_Name, O_Contact, No_of_godown) VALUES (%s, %s, %s)",
                       (o_name, o_contact, no_of_godown))
        print("Owner inserted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

# Insertion function for Godown table
def godown_insertion(conn, cursor):
    print("Inserting into Godown table...")
    g_city = input("Enter godown city: ")
    g_state = input("Enter godown state: ")
    capacity = int(input("Enter godown capacity: "))
    g_type = input("Enter godown type: ")
    owner_id = int(input("Enter owner ID: "))

    # Check if owner exists
    cursor.execute("SELECT Owner_id FROM owner WHERE Owner_id = %s", (owner_id,))
    result = cursor.fetchone()
    if result is None:
        print("Error: Owner with ID {} does not exist.".format(owner_id))
        return

    # Insert into Godown table
    try:
        cursor.execute("INSERT INTO Godown (G_city, G_state, capacity, G_type, Owner_id)
VALUES (%s, %s, %s, %s, %s)",
                       (g_city, g_state, capacity, g_type, owner_id))
        print("Godown inserted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

# Insertion function for Employee table
def employee_insertion(conn, cursor):
    print("Inserting into Employee table...")
    e_name = input("Enter employee name: ")
    e_contact = input("Enter employee contact: ")
    salary = int(input("Enter employee salary: "))
    joining_date = input("Enter employee joining date (YYYY-MM-DD): ")
    working_godown = int(input("Enter working godown ID: "))

    # Check if godown exists
    cursor.execute("SELECT Godown_id FROM Godown WHERE Godown_id = %s",
(working_godown,))
    result = cursor.fetchone()
    if result is None:
        print("Error: Godown with ID {} does not exist.".format(working_godown))
        return

    # Insert into Employee table
    try:
        cursor.execute("INSERT INTO employee (E_Name, E_Contact, Salary, joining_date,
Working_Godown) VALUES (%s, %s, %s, %s, %s)",
                       (e_name, e_contact, salary, joining_date, working_godown))
        print("Employee inserted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

# Insertion function for Farmer table
def farmer_insertion(conn, cursor):
    print("Inserting into Farmer table...")
    f_name = input("Enter farmer name: ")
    f_contact = input("Enter farmer contact: ")
    f_city = input("Enter farmer city: ")
    f_state = input("Enter farmer state: ")

    # Insert into Farmer table
    try:
        cursor.execute("INSERT INTO Farmer (F_Name, F_Contact, F_City, F_state) VALUES
(%s, %s, %s, %s)",
                       (f_name, f_contact, f_city, f_state))
        print("Farmer inserted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

# Insertion function for Product table
def product_insertion(conn, cursor):
    print("Inserting into Product table...")
    p_name = input("Enter product name: ")
    p_type = input("Enter product type: ")
    quantity = int(input("Enter product quantity: "))
    producer_id = int(input("Enter producer (farmer) ID: "))

    # Check if farmer exists
    cursor.execute("SELECT Farmer_id FROM Farmer WHERE Farmer_id = %s",
(producer_id,))
    result = cursor.fetchone()
    if result is None:
        print("Error: Farmer with ID {} does not exist.".format(producer_id))
        return

    # Insert into Product table
    try:
        cursor.execute("INSERT INTO Product (P_Name, P_type, quantity, producer) VALUES
(%s, %s, %s, %s)",
                       (p_name, p_type, quantity, producer_id))
        print("Product inserted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def booking_insertion(conn, cursor):
    print("Inserting into Booking table...")
    product_id = int(input("Enter product ID: "))
    godown_id = int(input("Enter godown ID: "))

    # Check if product exists
    cursor.execute("SELECT Product_id, quantity FROM Product WHERE Product_id = %s",
(product_id,))
    result_product = cursor.fetchone()
    if result_product is None:
        print("Error: Product with ID {} does not exist.".format(product_id))
        return
    product_quantity = result_product[1]

    # Check if godown exists
    cursor.execute("SELECT Godown_id, capacity FROM Godown WHERE Godown_id = %s",
(godown_id,))
    result_godown = cursor.fetchone()
    if result_godown is None:
        print("Error: Godown with ID {} does not exist.".format(godown_id))
        return
    godown_capacity = result_godown[1]

    # Check if the product is already booked in the selected godown
    cursor.execute("SELECT * FROM Booking WHERE Product_id = %s AND Godown_id
= %s", (product_id, godown_id))
    result_booking = cursor.fetchone()
    if result_booking:
        print("Error: Product with ID {} is already booked in godown with ID
{}.".format(product_id, godown_id))
        return

    # Check if there is enough capacity in the godown
    cursor.execute("SELECT SUM(quantity) FROM Booking JOIN Product ON
Booking.Product_id = Product.Product_id WHERE Godown_id = %s", (godown_id,))
    result_booked_quantity = cursor.fetchone()
    booked_quantity = result_booked_quantity[0] if result_booked_quantity[0] else 0

    if product_quantity + booked_quantity > godown_capacity:
        print("Error: Not enough capacity in godown with ID {}.".format(godown_id))
        return

    # Insert into Booking table
    try:
        cursor.execute("INSERT INTO Booking (Product_id, Godown_id) VALUES (%s, %s)",
                       (product_id, godown_id))
        print("Booking inserted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)



# Function for deletion
def deletion_menu(conn):
    cursor = conn.cursor()

    print("1. Delete from Owner table")
    print("2. Delete from Godown table")
    print("3. Delete from Employee table")
    print("4. Delete from Farmer table")
    print("5. Delete from Product table")
    print("6. Delete from Booking table")

    choice = input("Enter your choice: ")

    if choice == "1":
        owner_deletion(conn, cursor)
    elif choice == "2":
        godown_deletion(conn, cursor)
    elif choice == "3":
        employee_deletion(conn, cursor)
    elif choice == "4":
        farmer_deletion(conn, cursor)
    elif choice == "5":
        product_deletion(conn, cursor)
    elif choice == "6":
        booking_deletion(conn, cursor)
    else:
        print("Invalid choice.")

    conn.commit()
    cursor.close()
```

```
def owner_deletion(conn, cursor):
    print("Deleting from Owner table...")
    owner_id = int(input("Enter owner ID to delete: "))

    try:
        # Get the IDs of godowns owned by the owner
        cursor.execute("SELECT Godown_id FROM Godown WHERE Owner_id = %s",
(owner_id,))
        godown_ids = [row[0] for row in cursor.fetchall()]

        # Delete bookings associated with the godowns
        if godown_ids:
            placeholders = ', '.join(['%s'] * len(godown_ids))
            cursor.execute("DELETE FROM Booking WHERE Godown_id IN
({})".format(placeholders), godown_ids)

        # Delete employees working in the godowns
        if godown_ids:
            placeholders = ', '.join(['%s'] * len(godown_ids))
            cursor.execute("DELETE FROM Employee WHERE Working_Godown IN
({})".format(placeholders), godown_ids)

        # Delete godowns
        cursor.execute("DELETE FROM Godown WHERE Owner_id = %s", (owner_id,))

        # Delete owner
        cursor.execute("DELETE FROM Owner WHERE Owner_id = %s", (owner_id,))

        print("Owner and related records deleted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)



def godown_deletion(conn, cursor):
    print("Deleting from Godown table...")
    godown_id = int(input("Enter godown ID to delete: "))

    try:
        # Perform cascading delete in child tables
        cursor.execute("DELETE FROM Booking WHERE Godown_id = %s", (godown_id,))
        cursor.execute("DELETE FROM Godown WHERE Godown_id = %s", (godown_id,))
        print("Godown and related records deleted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def employee_deletion(conn, cursor):
    print("Deleting from Employee table...")
    employee_id = int(input("Enter employee ID to delete: "))

    try:
        # Perform cascading delete in child tables
        cursor.execute("DELETE FROM employee WHERE Employee_id = %s", (employee_id,))
        print("Employee deleted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def farmer_deletion(conn, cursor):
    print("Deleting from Farmer table...")
    farmer_id = int(input("Enter farmer ID to delete: "))

    try:
        # Delete products of the farmer from Booking table
        cursor.execute("DELETE FROM Booking WHERE Product_id IN (SELECT Product_id
FROM Product WHERE producer = %s)", (farmer_id,))

        # Perform cascading delete in child tables
        cursor.execute("DELETE FROM Product WHERE producer = %s", (farmer_id,))
        cursor.execute("DELETE FROM Farmer WHERE Farmer_id = %s", (farmer_id,))

        print("Farmer, related products, and bookings deleted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def product_deletion(conn, cursor):
    print("Deleting from Product table...")
    product_id = int(input("Enter product ID to delete: "))

    try:
        # Perform cascading delete in child tables
        cursor.execute("DELETE FROM Booking WHERE Product_id = %s", (product_id,))
        cursor.execute("DELETE FROM Product WHERE Product_id = %s", (product_id,))
        print("Product and related records deleted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)
```

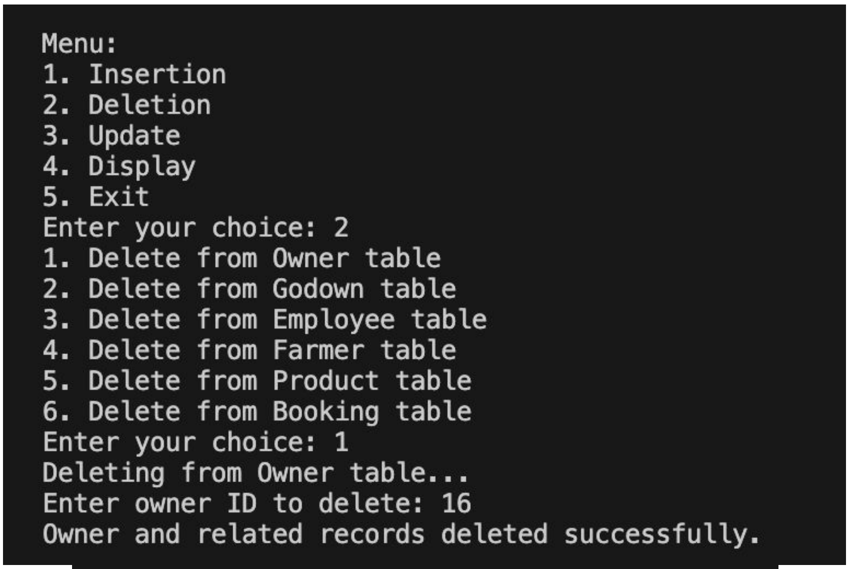
```
def booking_deletion(conn, cursor):
    print("Deleting from Booking table...")
    booking_id = int(input("Enter booking ID to delete: "))

    try:
        # Perform cascading delete in child tables
        cursor.execute("DELETE FROM Booking WHERE Booking_id = %s", (booking_id,))
        print("Booking deleted successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)


def product_update(conn, cursor):
    print("Updating Product table...")
    product_id = int(input("Enter product ID to update: "))
    new_name = input("Enter new product name: ")
    new_type = input("Enter new product type: ")
    new_quantity = int(input("Enter new product quantity: "))
    new_producer_id = int(input("Enter new producer (farmer) ID: "))

    # Check if farmer exists
    cursor.execute("SELECT Farmer_id FROM Farmer WHERE Farmer_id = %s",
(new_producer_id,))
    result = cursor.fetchone()
    if result is None:
        print("Error: Farmer with ID {} does not exist.".format(new_producer_id))
        return

    # Update Product table
    try:
        cursor.execute("UPDATE Product SET P_Name = %s, P_type = %s, quantity = %s,
producer = %s WHERE Product_id = %s",
                       (new_name, new_type, new_quantity, new_producer_id, product_id))
        print("Product updated successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def booking_update(conn, cursor):
    print("Updating Booking table...")
    booking_id = int(input("Enter booking ID to update: "))
    new_product_id = int(input("Enter new product ID: "))
    new_godown_id = int(input("Enter new godown ID: "))
    # Check if product exists
    cursor.execute("SELECT Product_id FROM Product WHERE Product_id = %s",
(new_product_id,))
    result_product = cursor.fetchone()
    if result_product is None:
        print("Error: Product with ID {} does not exist.".format(new_product_id))
        return

    # Check if godown exists
    cursor.execute("SELECT Godown_id FROM Godown WHERE Godown_id = %s",
(new_godown_id,))
    result_godown = cursor.fetchone()
    if result_godown is None:
        print("Error: Godown with ID {} does not exist.".format(new_godown_id))
        return

    # Update Booking table
    try:
        cursor.execute("UPDATE Booking SET Product_id = %s, Godown_id = %s WHERE
Booking_id = %s",
                       (new_product_id, new_godown_id, booking_id))
        print("Booking updated successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)


# Function for update
def update_menu(conn):
    cursor = conn.cursor()

    print("1. Update Owner table")
    print("2. Update Godown table")
    print("3. Update Employee table")
    print("4. Update Farmer table")
    print("5. Update Product table")
    print("6. Update Booking table")

    choice = input("Enter your choice: ")

    if choice == "1":
        owner_update(conn, cursor)
    elif choice == "2":
        godown_update(conn, cursor)
    elif choice == "3":
        employee_update(conn, cursor)
    elif choice == "4":
        farmer_update(conn, cursor)
    elif choice == "5":
        product_update(conn, cursor)
    elif choice == "6":
        booking_update(conn, cursor)
    else:
        print("Invalid choice.")

    conn.commit()
    cursor.close()

# Update functions for each table
def owner_update(conn, cursor):
    print("Updating Owner table...")
    owner_id = int(input("Enter owner ID to update: "))
    new_name = input("Enter new owner name: ")
    new_contact = input("Enter new owner contact: ")

    # Update Owner table
    try:
        cursor.execute("UPDATE Owner SET O_Name = %s, O_Contact = %s WHERE
Owner_id = %s",
                       (new_name, new_contact, owner_id))
        print("Owner updated successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def godown_update(conn, cursor):
    print("Updating Godown table...")
    godown_id = int(input("Enter godown ID to update: "))
    new_city = input("Enter new city: ")
    new_state = input("Enter new state: ")
    new_capacity = int(input("Enter new capacity: "))
    new_type = input("Enter new type: ")

    # Update Godown table
    try:
        cursor.execute("UPDATE Godown SET G_city = %s, G_state = %s, capacity = %s,
G_type = %s WHERE Godown_id = %s",
                       (new_city, new_state, new_capacity, new_type, godown_id))
        print("Godown updated successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def employee_update(conn, cursor):
    print("Updating Employee table...")
    employee_id = int(input("Enter employee ID to update: "))
    new_name = input("Enter new employee name: ")
    new_contact = input("Enter new employee contact: ")
    new_salary = int(input("Enter new employee salary: "))
    new_joining_date = input("Enter new joining date (YYYY-MM-DD): ")

    # Update Employee table
    try:
        cursor.execute("UPDATE Employee SET E_Name = %s, E_Contact = %s, Salary = %s,
joining_date = %s WHERE Employee_id = %s",
                      (new_name, new_contact, new_salary, new_joining_date, employee_id))
        print("Employee updated successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)

def farmer_update(conn, cursor):
    print("Updating Farmer table...")
    farmer_id = int(input("Enter farmer ID to update: "))
    new_name = input("Enter new farmer name: ")
    new_contact = input("Enter new farmer contact: ")
    new_city = input("Enter new farmer city: ")
    new_state = input("Enter new farmer state: ")
    # Update Farmer table
    try:
        cursor.execute("UPDATE Farmer SET F_Name = %s, F_Contact = %s, F_City = %s,
F_state = %s WHERE Farmer_id = %s",
                    (new_name, new_contact, new_city, new_state, farmer_id))
        print("Farmer updated successfully.")
    except mysql.connector.Error as err:
        print("Error:", err)


def display_table_menu(conn):
    cursor = conn.cursor()

    while True:
        print("\nMenu:")
        print("1. Display Owner table")
        print("2. Display Godown table")
        print("3. Display Employee table")
        print("4. Display Farmer table")
        print("5. Display Product table")
        print("6. Display Booking table")
        print("0. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            display_table(conn, "Owner")
        elif choice == "2":
            display_table(conn, "Godown")
        elif choice == "3":
            display_table(conn, "Employee")
        elif choice == "4":
            display_table(conn, "Farmer")
        elif choice == "5":
            display_table(conn, "Product")
        elif choice == "6":
            display_table(conn, "Booking")
        elif choice == "0":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

    cursor.close()

def display_table(conn, table_name):
    cursor = conn.cursor()
    try:
        cursor.execute("SELECT * FROM {}".format(table_name))
        columns = [col[0] for col in cursor.description]
        rows = cursor.fetchall()
        if rows:
            print("\n{} Table:".format(table_name))
            print("-" * 30)
            for col in columns:
                print(col, end='\t')
            print("\n" + "-" * 30)
            for row in rows:
                for val in row:
                    print(val, end='\t')
                print()
            print("-" * 30)
        else:
            print("No data found in {} table.".format(table_name))
    except mysql.connector.Error as err:
        print("Error:", err)


# Main function
def main():
    conn = connect_to_database()
    if conn is None:
        return

    logged_in = False
    while not logged_in:
        logged_in = login()

    while True:
        print("\nMenu:")
        print("1. Insertion")
        print("2. Deletion")
        print("3. Update")
        print("4. Display")
        print("5. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            insertion_menu(conn)
        elif choice == "2":
            deletion_menu(conn)
        elif choice == "3":
            update_menu(conn)
        elif choice == "4":
            display_table_menu(conn)
        elif (choice == "5"):
            break
        else:
            print("Invalid choice. Please try again.")

    conn.close()
```

main()

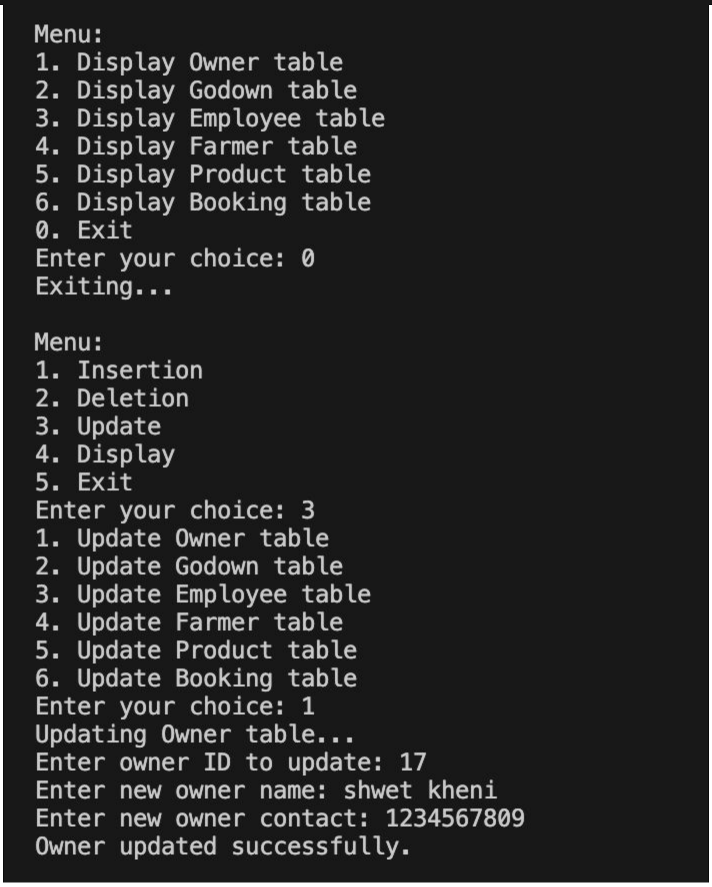
Outputs



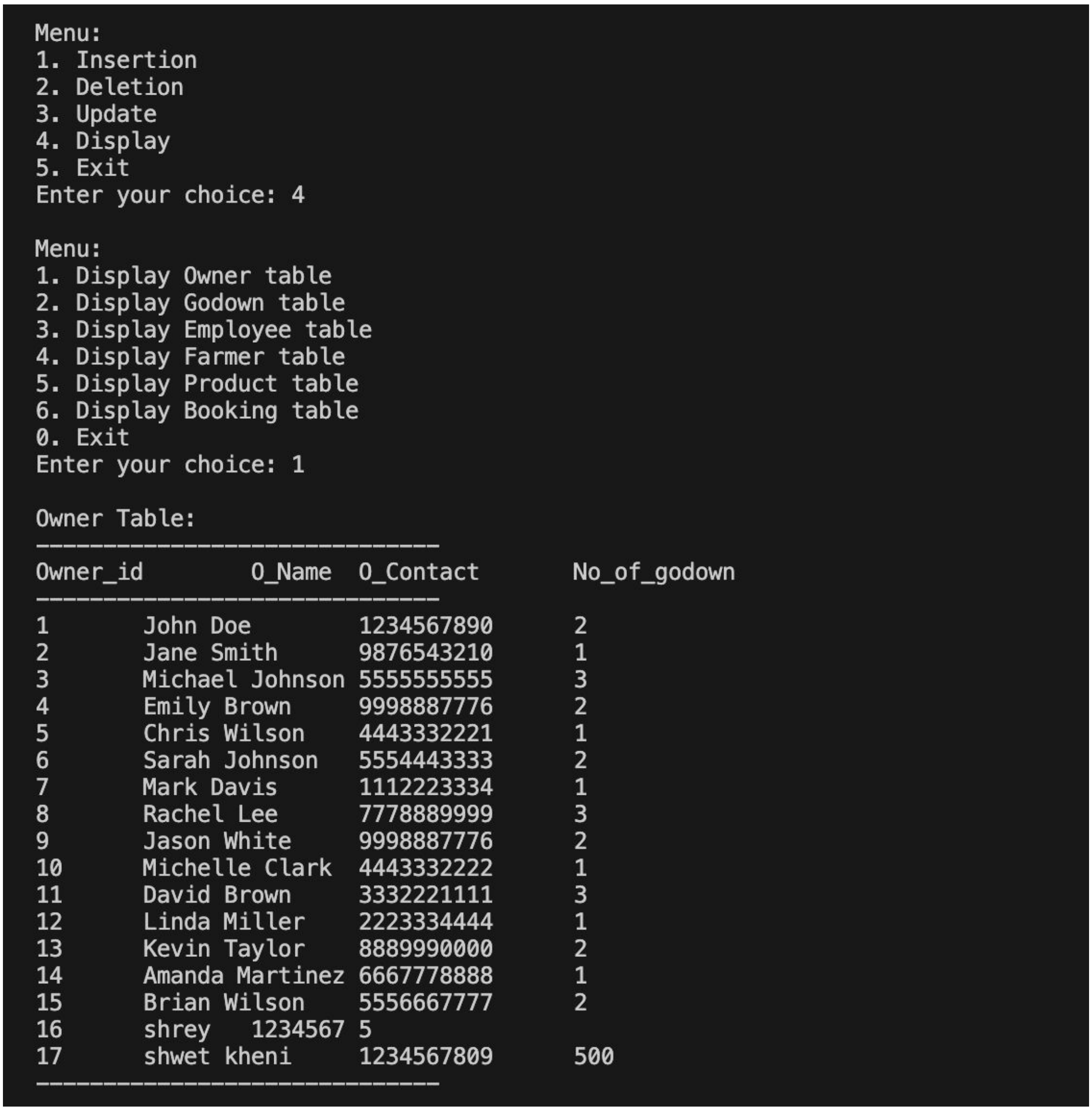
```
Product Table:
------------------------------
Product_id      P_Name  P_type  quantity        producer
------------------------------
1       Wheat   Grain   500     1
2       Apples  Fruit   300     2
3       Milk    Dairy   200     3
4       Corn    Grain   400     4
5       Chicken Meat    250     5
6       Rice    Grain   600     6
7       Oranges Fruit   350     7
8       Eggs    Dairy   250     8
9       Barley  Grain   450     9
10      Beef    Meat    300     10
11      Tomatoes        Vegetable       400     11
12      Cheese  Dairy   280     12
13      Potatoes        Vegetable       350     13
14      Pork    Meat    280     14
15      Bananas Fruit   400     15
16      xyz     xyz     4       15
17      ert     ert     1000    15
------------------------------

Menu:
1. Display Owner table
2. Display Godown table
3. Display Employee table
4. Display Farmer table
5. Display Product table
6. Display Booking table
0. Exit
Enter your choice: 0
Exiting...

Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 1
1. Insert into Owner table
2. Insert into Godown table
3. Insert into Employee table
4. Insert into Farmer table
5. Insert into Product table
6. Insert into Booking table
Enter your choice: 6
Inserting into Booking table...
Enter product ID: 17
Enter godown ID: 14
Error: Not enough capacity in godown with ID 14.
```



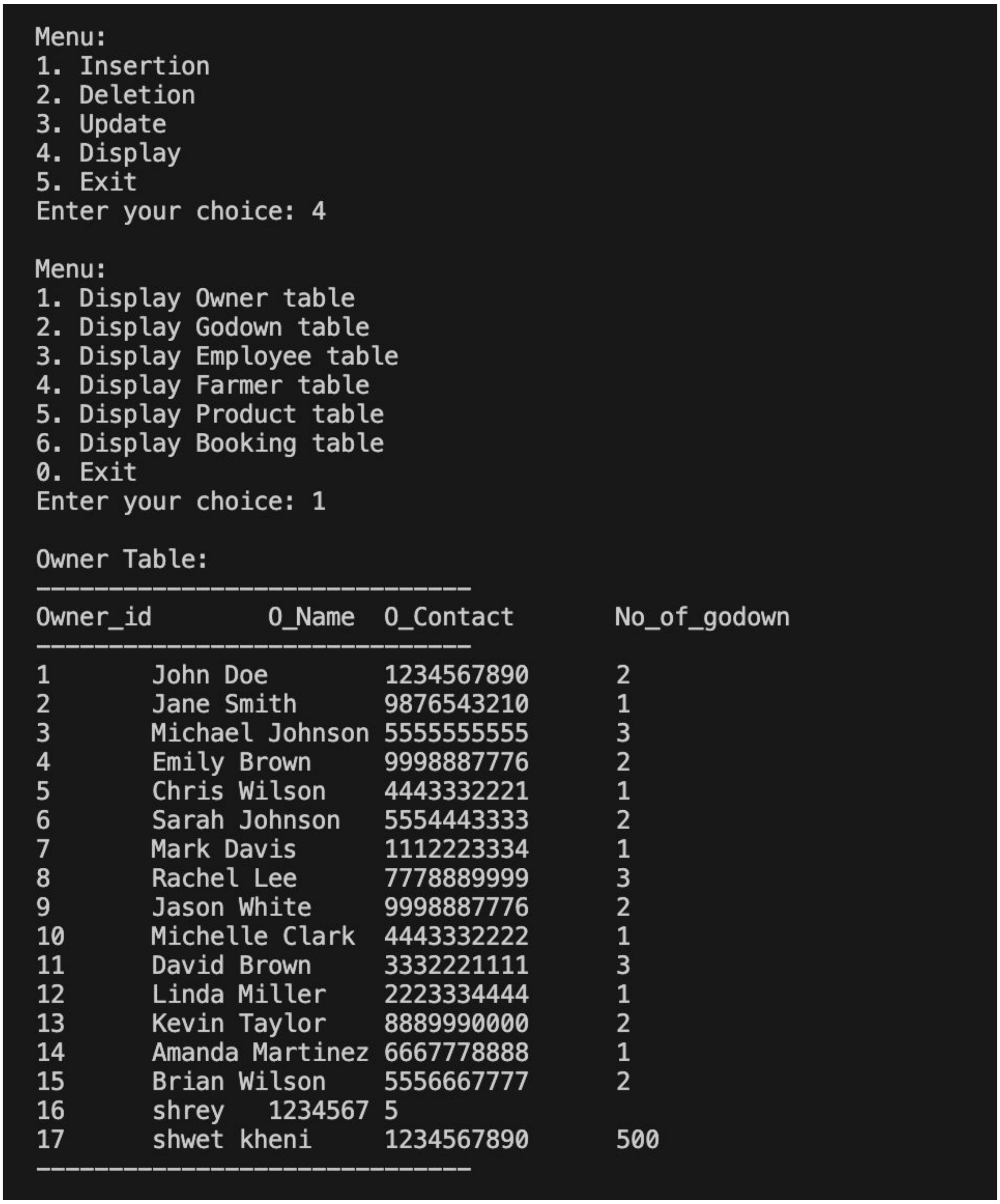
```
Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 2
1. Delete from Owner table
2. Delete from Godown table
3. Delete from Employee table
4. Delete from Farmer table
5. Delete from Product table
6. Delete from Booking table
Enter your choice: 1
Deleting from Owner table...
Enter owner ID to delete: 16
Owner and related records deleted successfully.
```



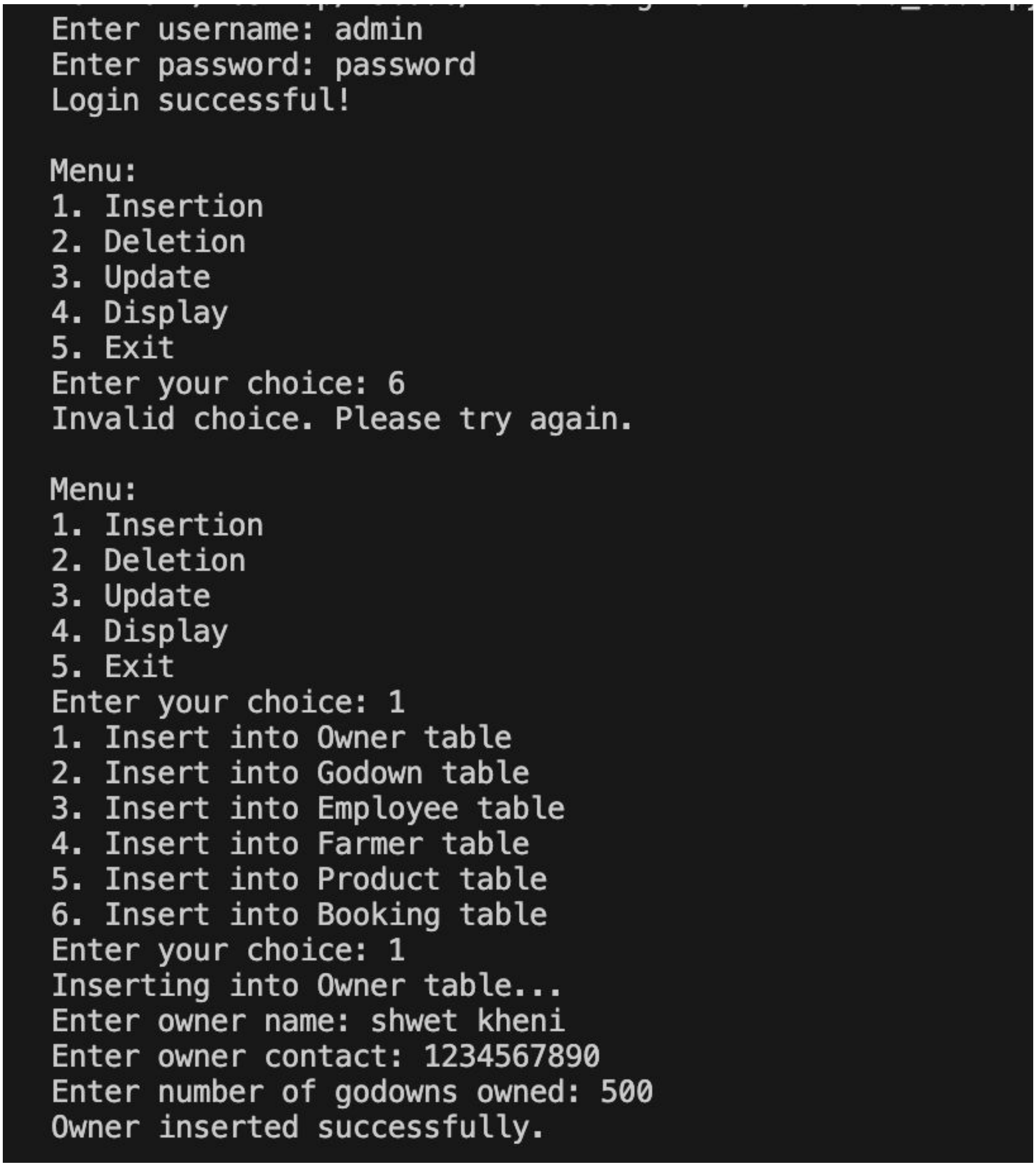
```
Menu:
1. Display Owner table
2. Display Godown table
3. Display Employee table
4. Display Farmer table
5. Display Product table
6. Display Booking table
0. Exit
Enter your choice: 0
Exiting...

Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 3
1. Update Owner table
2. Update Godown table
3. Update Employee table
4. Update Farmer table
5. Update Product table
6. Update Booking table
Enter your choice: 1
Updating Owner table...
Enter owner ID to update: 17
Enter new owner name: shwet kheni
Enter new owner contact: 1234567809
Owner updated successfully.
```



```
Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 4

Menu:
1. Display Owner table
2. Display Godown table
3. Display Employee table
4. Display Farmer table
5. Display Product table
6. Display Booking table
0. Exit
Enter your choice: 1

Owner Table:
------------------------------
Owner_id        O_Name  O_Contact       No_of_godown
------------------------------
1       John Doe        1234567890      2
2       Jane Smith      9876543210      1
3       Michael Johnson 5555555555      3
4       Emily Brown     9998887776      2
5       Chris Wilson    4443332221      1
6       Sarah Johnson   5554443333      2
7       Mark Davis      1112223334      1
8       Rachel Lee      7778889999      3
9       Jason White     9998887776      2
10      Michelle Clark  4443332222      1
11      David Brown     3332221111      3
12      Linda Miller    2223334444      1
13      Kevin Taylor    8889990000      2
14      Amanda Martinez 6667778888      1
15      Brian Wilson    5556667777      2
16      shrey   1234567 5
17      shwet kheni     1234567809      500
------------------------------
```



```
Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 4

Menu:
1. Display Owner table
2. Display Godown table
3. Display Employee table
4. Display Farmer table
5. Display Product table
6. Display Booking table
0. Exit
Enter your choice: 1

Owner Table:
------------------------------
Owner_id        O_Name  O_Contact       No_of_godown
------------------------------
1       John Doe        1234567890      2
2       Jane Smith      9876543210      1
3       Michael Johnson 5555555555      3
4       Emily Brown     9998887776      2
5       Chris Wilson    4443332221      1
6       Sarah Johnson   5554443333      2
7       Mark Davis      1112223334      1
8       Rachel Lee      7778889999      3
9       Jason White     9998887776      2
10      Michelle Clark  4443332222      1
11      David Brown     3332221111      3
12      Linda Miller    2223334444      1
13      Kevin Taylor    8889990000      2
14      Amanda Martinez 6667778888      1
15      Brian Wilson    5556667777      2
16      shrey   1234567 5
17      shwet kheni     1234567890      500
------------------------------
```



```
Enter username: admin
Enter password: password
Login successful!

Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 6
Invalid choice. Please try again.

Menu:
1. Insertion
2. Deletion
3. Update
4. Display
5. Exit
Enter your choice: 1
1. Insert into Owner table
2. Insert into Godown table
3. Insert into Employee table
4. Insert into Farmer table
5. Insert into Product table
6. Insert into Booking table
Enter your choice: 1
Inserting into Owner table...
Enter owner name: shwet kheni
Enter owner contact: 1234567890
Enter number of godowns owned: 500
Owner inserted successfully.
```