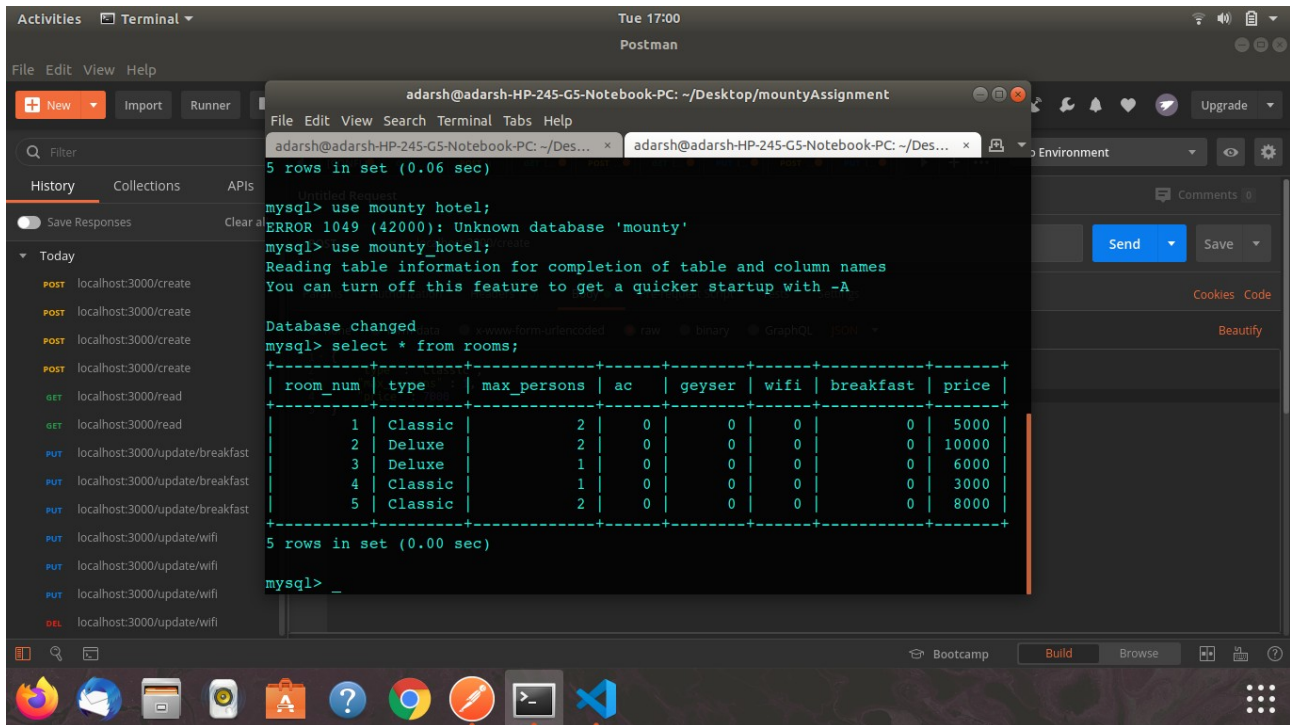


Assignment for backend developer role (Backend for managing a hotel info)

Add Route :

Add route adds new hotel rooms with details of the room that can further be updated. It can be easily demonstrated using the following screenshots.

Before adding a room

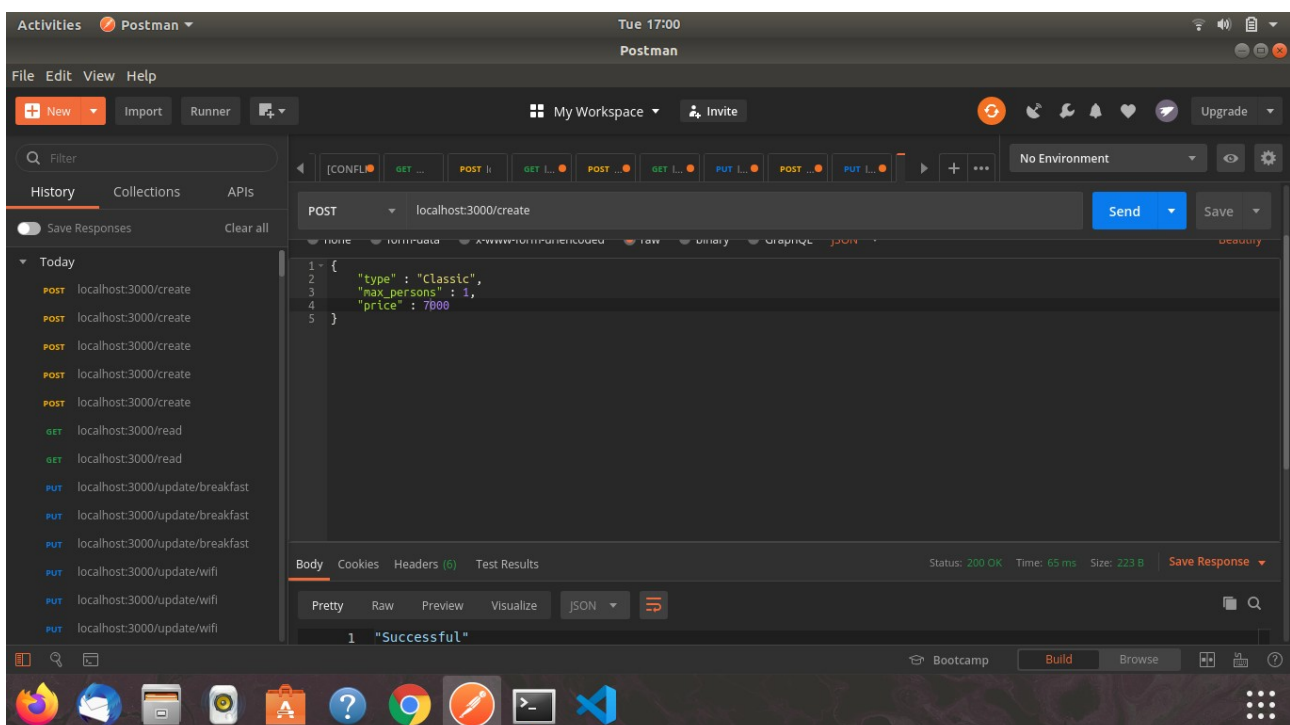


```
adarsh@adarsh-HP-245-G5-Notebook-PC: ~/Desktop/mountyAssignment
mysql> use mounty hotel;
ERROR 1049 (42000): Unknown database 'mounty'
mysql> use mounty_hotel;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from rooms;
+-----+-----+-----+-----+-----+-----+-----+-----+
| room_num | type   | max_persons | ac | geyser | wifi | breakfast | price |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Classic | 2 | 0 | 0 | 0 | 0 | 5000 |
| 2 | Deluxe | 2 | 0 | 0 | 0 | 0 | 10000 |
| 3 | Deluxe | 1 | 0 | 0 | 0 | 0 | 6000 |
| 4 | Classic | 1 | 0 | 0 | 0 | 0 | 3000 |
| 5 | Classic | 2 | 0 | 0 | 0 | 0 | 8000 |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> _
```

Adding a new room using postman :

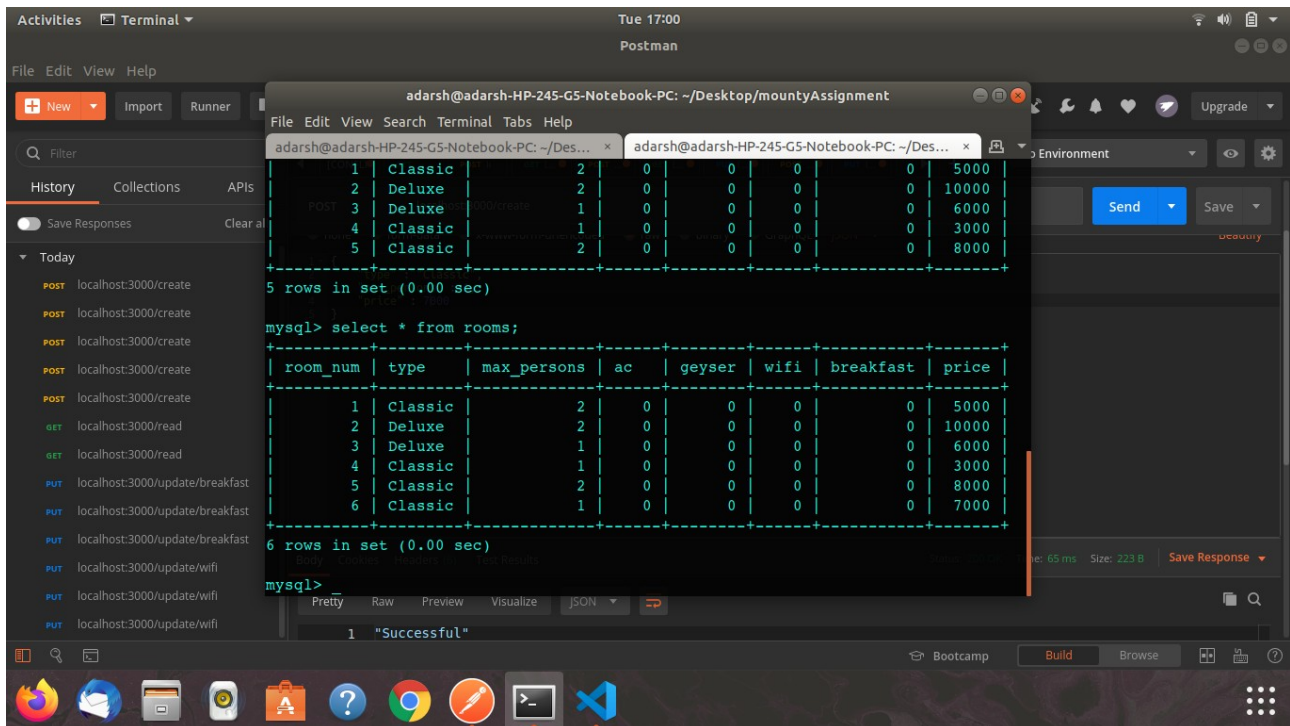


```
POST localhost:3000/create
{
  "type": "Classic",
  "max_persons": 1,
  "price": 7000
}
```

Status: 200 OK Time: 65 ms Size: 223 B Save Response

1 "Successful"

Updated table after the room gets added :



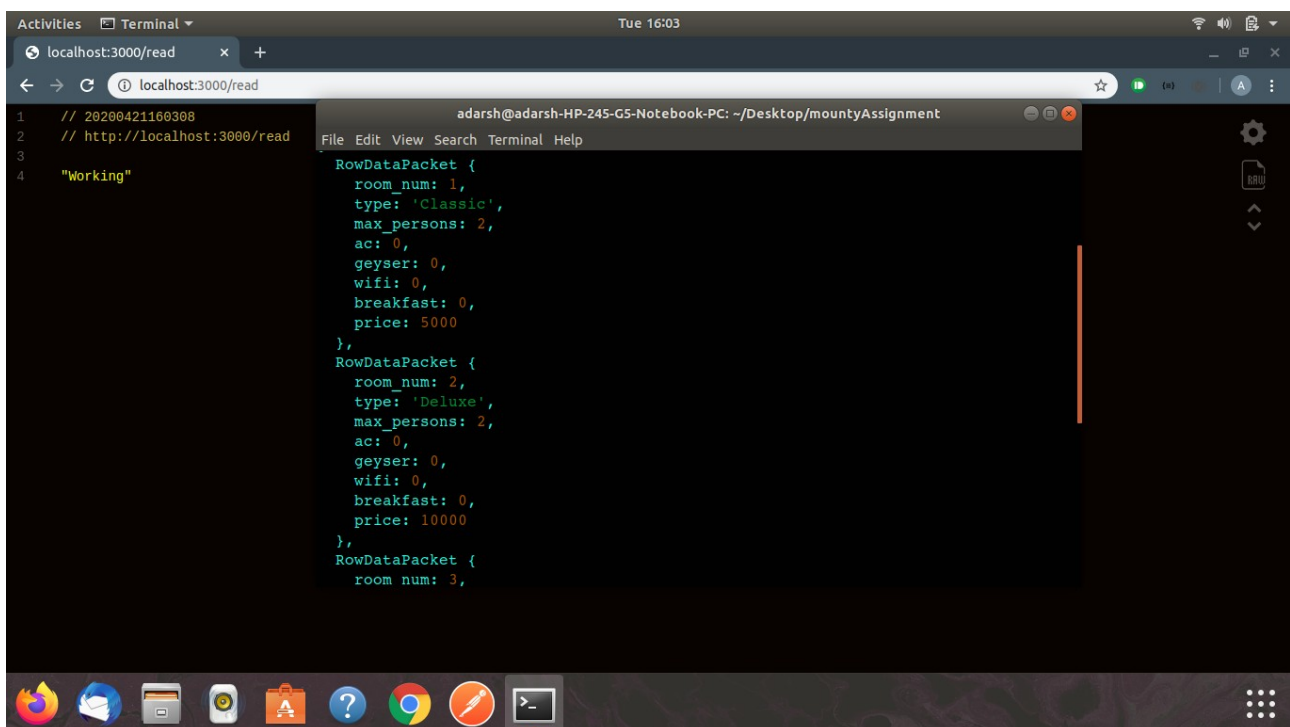
```
adarsh@adarsh-HP-245-G5-Notebook-PC: ~/Desktop/mountyAssignment
mysql> insert into rooms (room_num, type, max_persons, ac, geyser, wifi, breakfast, price) values (1, 'Classic', 2, 0, 0, 0, 0, 5000);
mysql> insert into rooms (room_num, type, max_persons, ac, geyser, wifi, breakfast, price) values (2, 'Deluxe', 2, 0, 0, 0, 0, 10000);
mysql> insert into rooms (room_num, type, max_persons, ac, geyser, wifi, breakfast, price) values (3, 'Deluxe', 1, 0, 0, 0, 0, 6000);
mysql> insert into rooms (room_num, type, max_persons, ac, geyser, wifi, breakfast, price) values (4, 'Classic', 1, 0, 0, 0, 0, 3000);
mysql> insert into rooms (room_num, type, max_persons, ac, geyser, wifi, breakfast, price) values (5, 'Classic', 2, 0, 0, 0, 0, 8000);
5 rows in set (0.00 sec)

mysql> select * from rooms;
+----+-----+-----+---+-----+-----+-----+-----+
| room_num | type   | max_persons | ac | geyser | wifi | breakfast | price |
+----+-----+-----+---+-----+-----+-----+-----+
| 1        | Classic | 2          | 0  | 0      | 0    | 0          | 5000  |
| 2        | Deluxe  | 2          | 0  | 0      | 0    | 0          | 10000 |
| 3        | Deluxe  | 1          | 0  | 0      | 0    | 0          | 6000  |
| 4        | Classic | 1          | 0  | 0      | 0    | 0          | 3000  |
| 5        | Classic | 2          | 0  | 0      | 0    | 0          | 8000  |
| 6        | Classic | 1          | 0  | 0      | 0    | 0          | 7000  |
+----+-----+-----+---+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Read Route :

It can be further updated for filtering rooms, but as of now it displays the information of all the rooms in the table(in the console).
Here's a view of it



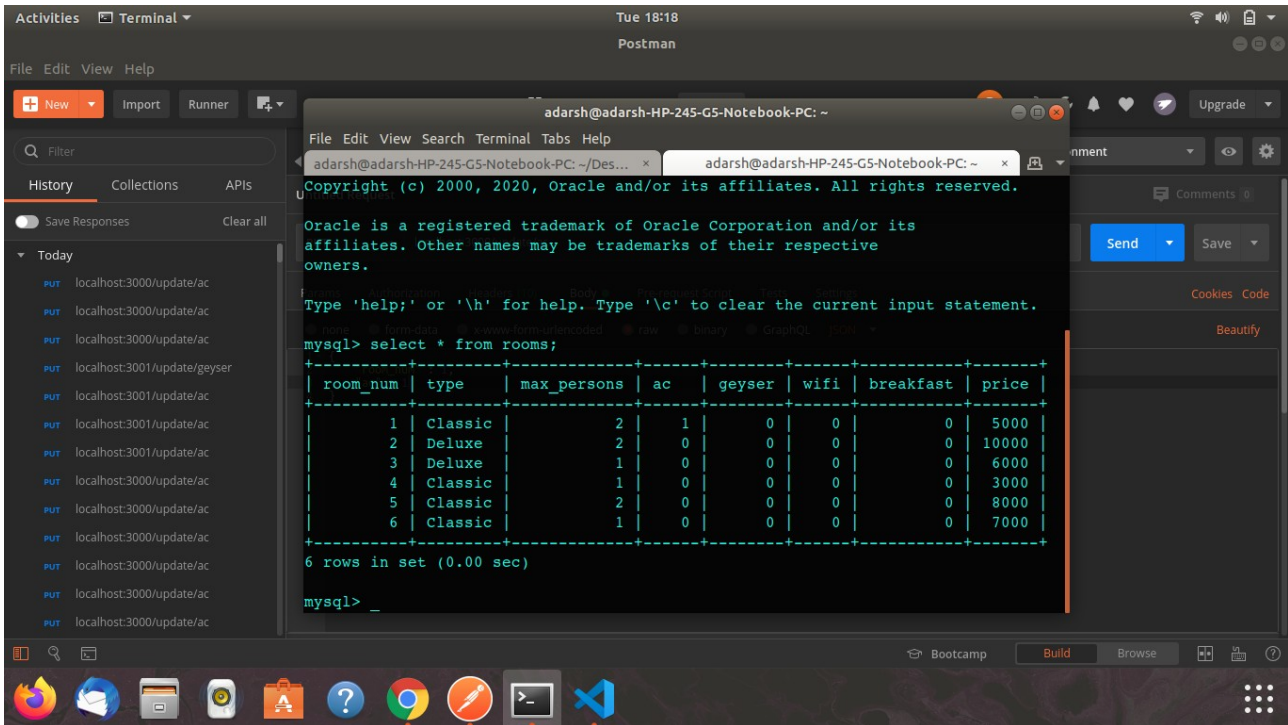
```
localhost:3000/read
// 20290421160308
// http://localhost:3000/read
"Working"

RowDataPacket {
  room_num: 1,
  type: 'Classic',
  max_persons: 2,
  ac: 0,
  geyser: 0,
  wifi: 0,
  breakfast: 0,
  price: 5000
},
RowDataPacket {
  room_num: 2,
  type: 'Deluxe',
  max_persons: 2,
  ac: 0,
  geyser: 0,
  wifi: 0,
  breakfast: 0,
  price: 10000
},
RowDataPacket {
  room_num: 3,
```

Update Route :

This route updates the details of the room that is already stored in the database of the hotel. For example if there was no AC in a room initially and the hotel members install AC in the room, the same can be updated via the update route. This route can update the information of AC, Geyser, Availability of wifi and inclusion of breakfast in a particular room. We can understand it via the following pasted screenshots.

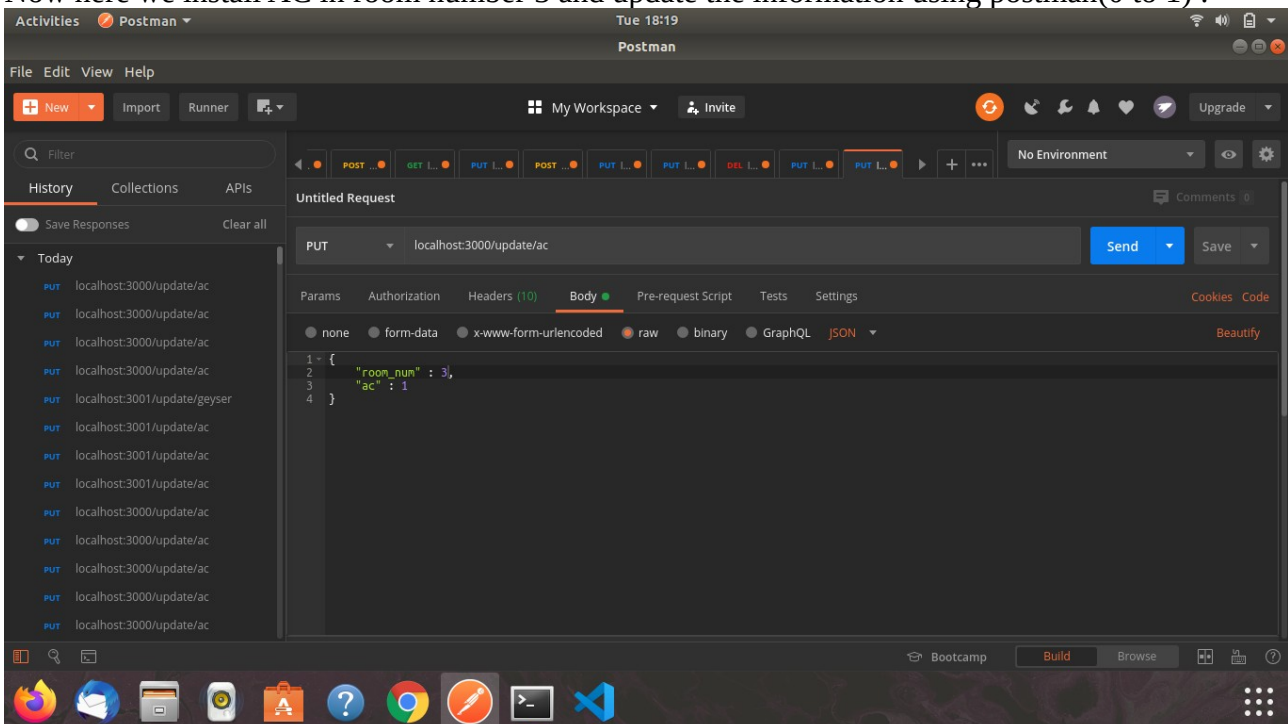
Initially room number 3 did not have AC in it (the AC attribute has value 0):



The screenshot shows a terminal window with a MySQL prompt. The query `mysql> select * from rooms;` has been executed, returning a table with 6 rows. The columns are `room_num`, `type`, `max_persons`, `ac`, `geyser`, `wifi`, `breakfast`, and `price`. Room 3 is a Deluxe room with 1 max person, 0 AC, 0 geyser, 0 wifi, 0 breakfast, and a price of 6000.

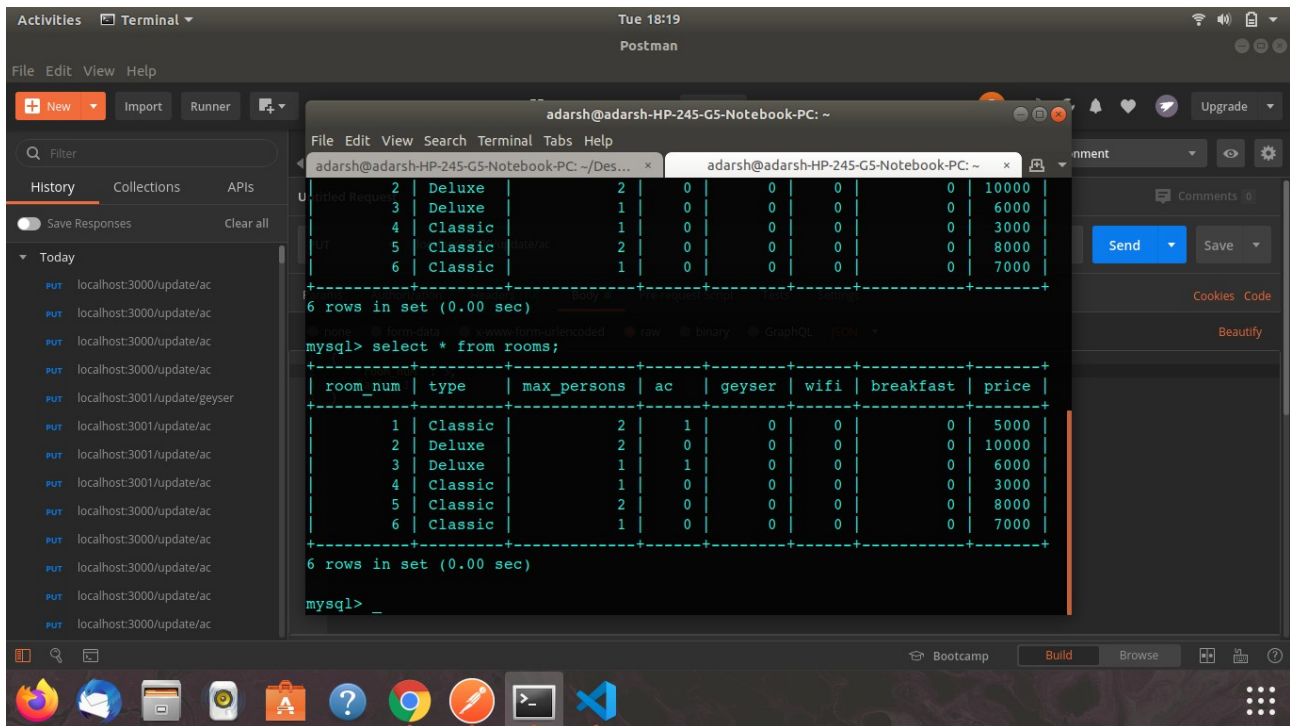
room_num	type	max_persons	ac	geyser	wifi	breakfast	price
1	Classic	2	1	0	0	0	5000
2	Deluxe	2	0	0	0	0	10000
3	Deluxe	1	0	0	0	0	6000
4	Classic	1	0	0	0	0	3000
5	Classic	2	0	0	0	0	8000
6	Classic	1	0	0	0	0	7000

Now here we install AC in room number 3 and update the information using postman(0 to 1) :



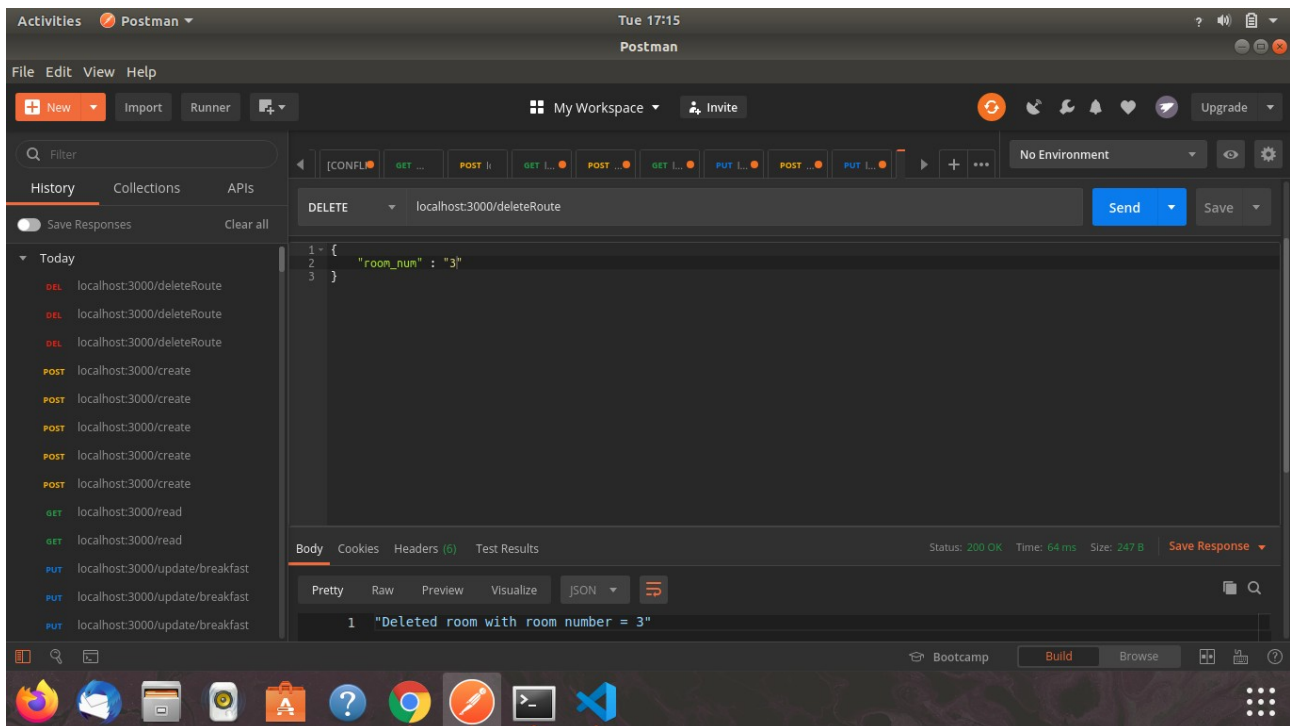
The screenshot shows the Postman application with a PUT request configured to `localhost:3000/update/ac`. The request body is a JSON object: `{ "room_num": 3, "ac": 1 }`. The interface shows the 'Body' tab with the JSON data.

After updating the information :



Delete Route :

Delete route can be used if the room is no longer available.



I hope it was helpful.

Thank you.