NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

DATABASE MANAGEMENT SYSTEM PROJECT REPORT

SEMESTER-3

MEMBERS
BALVINDER(1845)
OJAS(1825)
NIKITA(1803)
PULKIT(1804)

PROFESSOR
SHUSHMA NAGPAL

PROBLEM STATEMENT

The landBNB Online Booking system is a database system designed to facilitate the process of Booking properties like Flats, Villas and Hostels for staycation . The system will provide a set of features to access unique properties available in different branches of the aforementioned staycation chain in Delhi, and booking. The database will include a set of all branches in the Delhi region, a set of properties, a set of existing customers, a set of properties currently being selected by a customer and a set of all the invoices generated. Each customer will be identified by his/her customer_id. Upon launching the application, the users will be prompted to enter their phone number where if a match is found, they will be directed to the availability of the closest properties (according to the customer's selected address). Otherwise, the customer will be treated as a first time customer and will enter his/her information and then proceed towards the menu. Once a customer has registered he/she need not register again. The customer can choose from a variety of properties, in different locations and price ranges. The customer will be notified with the status of his/her booking and name of the host. The customer is expected to rate his/her experience upon (notification of) completion of the stay.

Requirement Analysis

We will be creating a database named landBnb.

To store the data we will create table to store data of Properties such as:
prop_id, prop_type(AC/NON AC), rent, no of beds and status(occupied/free).

To store the data we will create tables to store data of Customers such as:
Cust_id, id_proof, if_proof_number, no. of family members (male, female, children)

To store the data we will create tables to store data of Booking such as:
Booking_id, prop_id (to get data about the property), cust_id(to get data regarding the customer), doo (date of occupation), dol(date of leaving(bill generation)), rent_amount, advance_amount.

Once booking is done we can generate bill to store data about the bill we will use the existing booking table to store the data regarding billing information.

Many customers can book the same property at the same time so it is a many to one relationship.

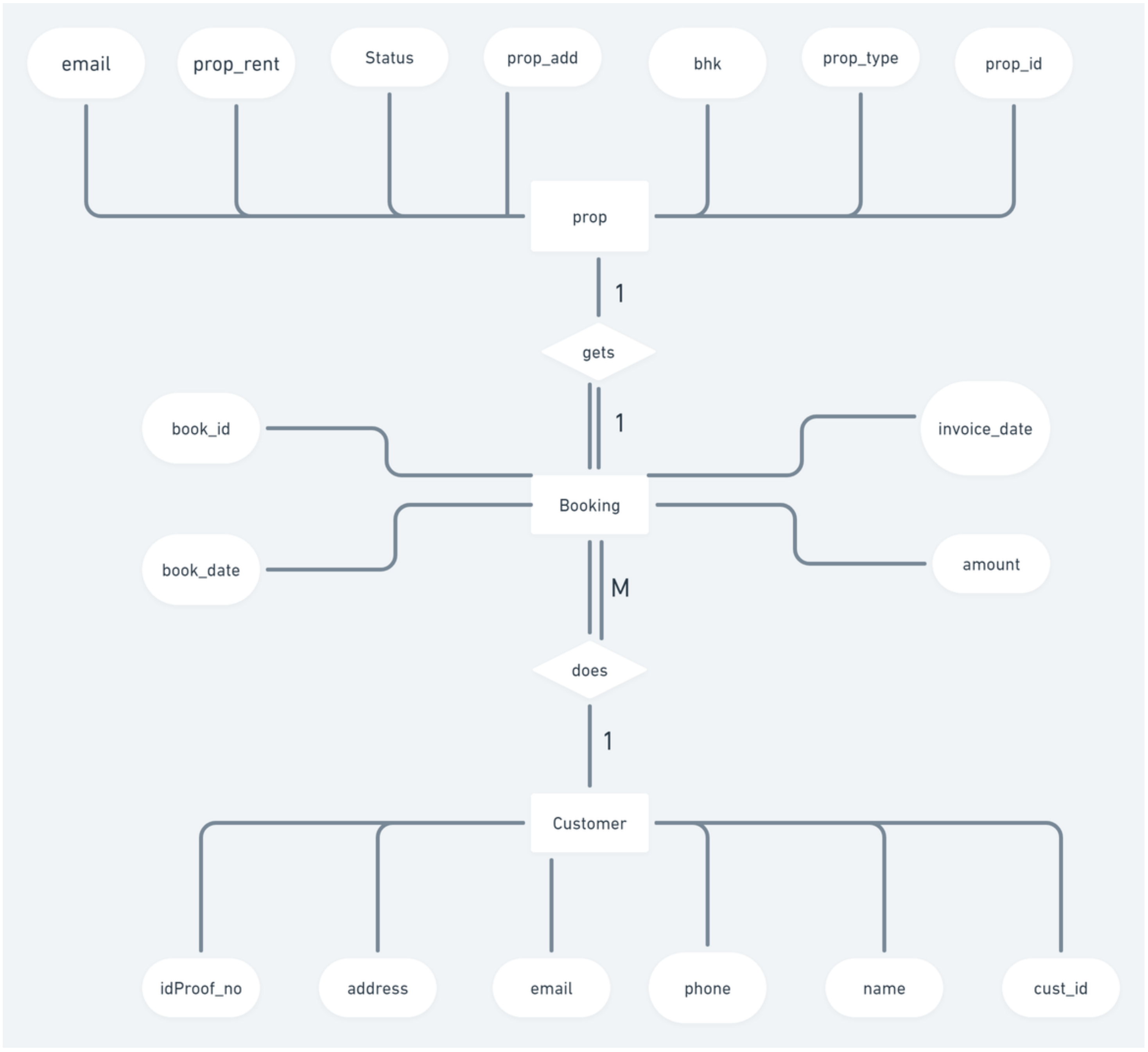
Each booking is made by a customer so there is a total participation of booking.

Each booking will be connected to only one of the property so there is one to one relationship.

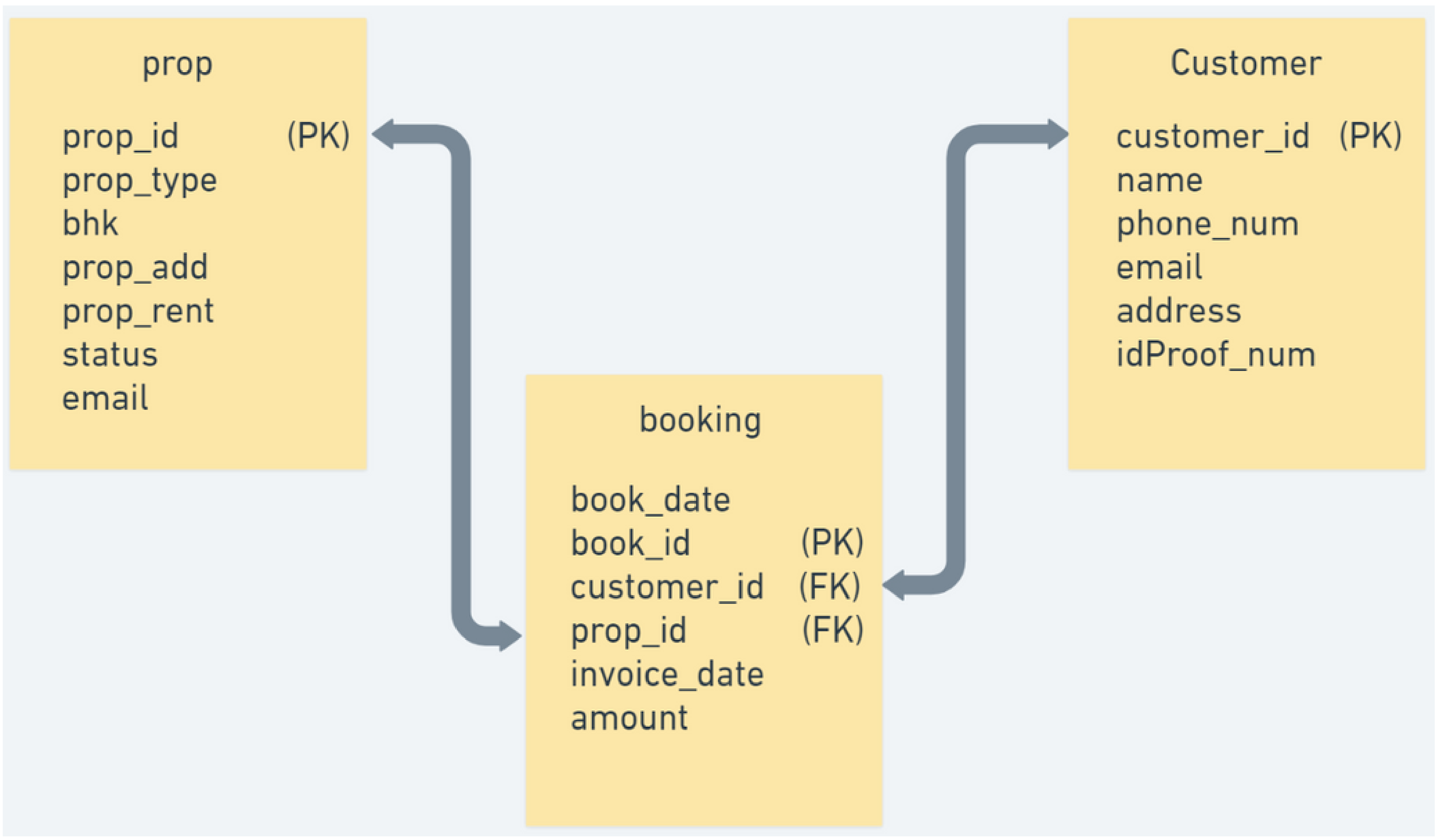
This way entire database can be created and connected.

ER Diagram

Link: https://whimsical.com/dbms-er-diagram-D1TpdeoWowpK9k2XE3YxPL



Relational Schema



BCNF

booking Table

book_id -> customer_id
book_id -> prop_id
book_id -> book_date
book_id -> invoice_date
book_id -> amount

‘book_id’ identifies all attributes thus servers as primary key as well as candidate key
And since it is a single attribute it is first, second, third and BCNF normal form

‘customer_id’ identifies all attributes thus servers as primary key as well as candidate key and since it is a single attribute it is first, second, third and BCNF normal form.

Customer Table

customer_id -> name
customer_id -> phone_num
customer_id -> email
customer_id -> address
customer_id -> idProof_num

prop Table

prop_id-> prop_type
prop_id-> bhk
prop_id-> prop_add
prop_id-> prop_rent
prop_id-> status
prop_id-> email

‘prop_id’ is a single attribute candidate key so ti is in first, second, third as well as BCNF normal form as neither attribute depends on part of candidate key nor attribute have any functional dependency as well.

Normalized Form

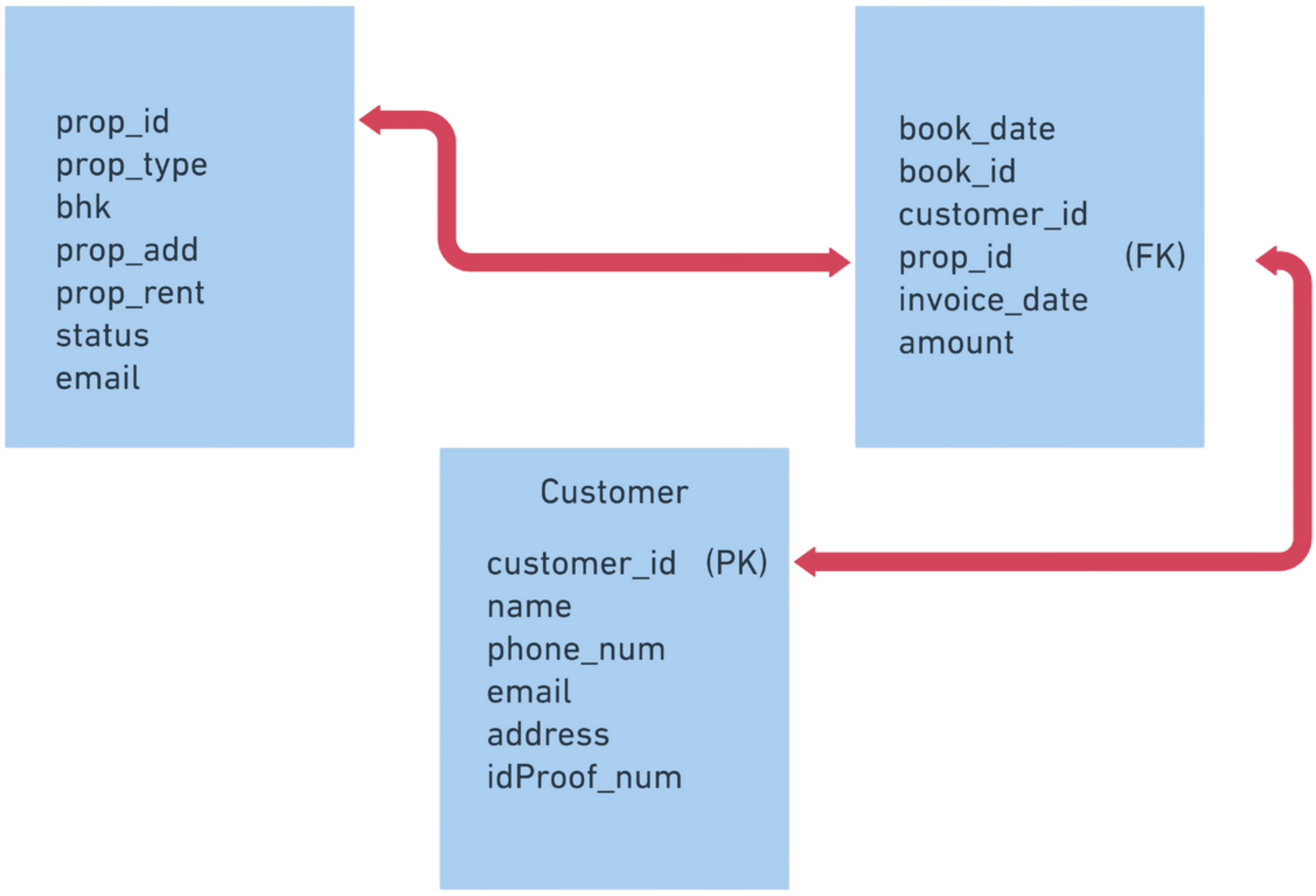
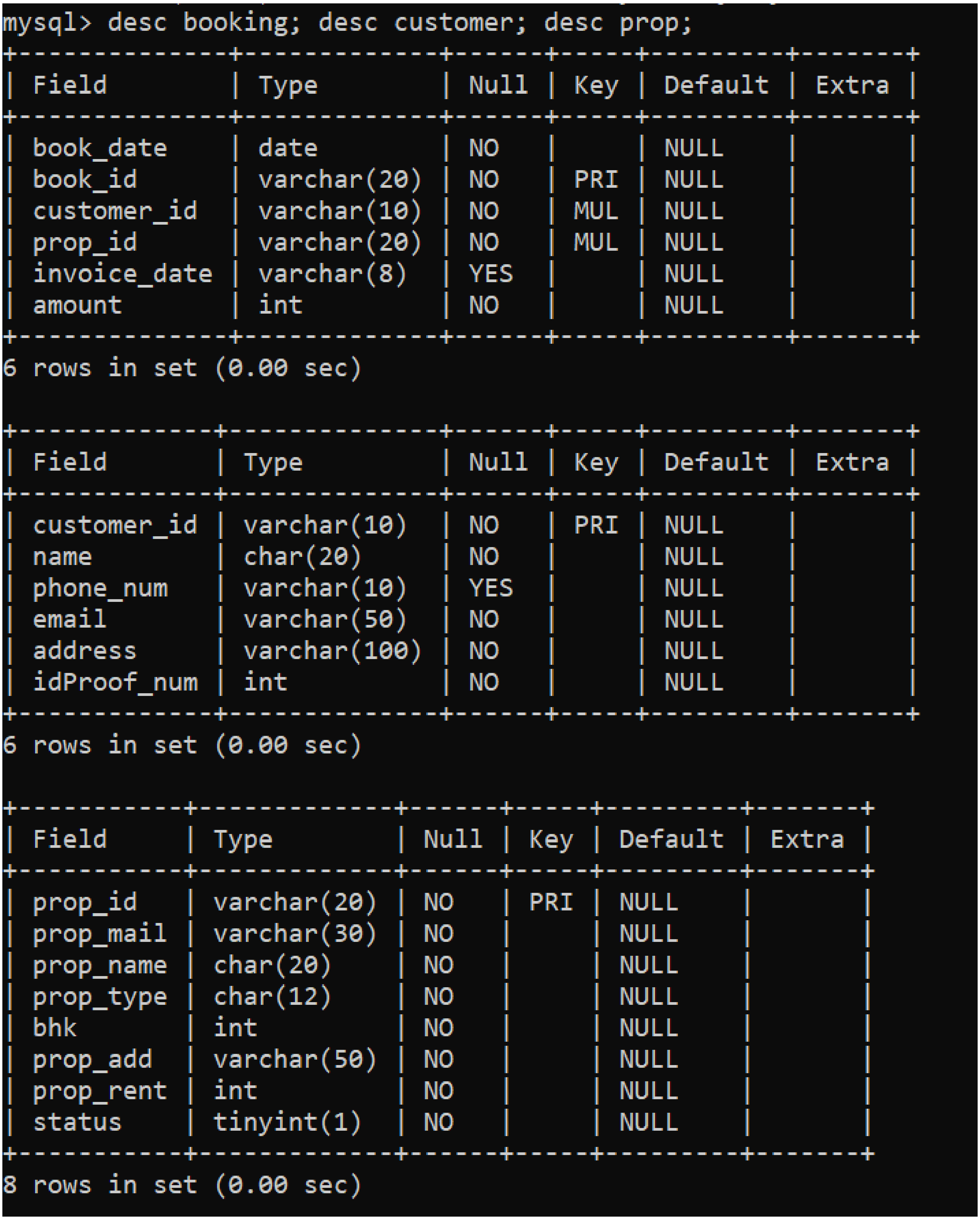


Table Structures



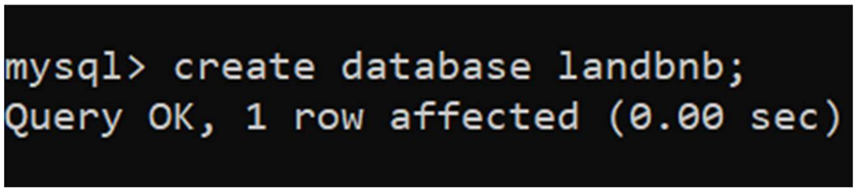
```
mysql> desc booking; desc customer; desc prop;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| book_date    | date        | NO   |     | NULL    |       |
| book_id      | varchar(20) | NO   | PRI | NULL    |       |
| customer_id  | varchar(10) | NO   | MUL | NULL    |       |
| prop_id      | varchar(20) | NO   | MUL | NULL    |       |
| invoice_date | varchar(8)  | YES  |     | NULL    |       |
| amount       | int         | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| customer_id | varchar(10)  | NO   | PRI | NULL    |       |
| name        | char(20)     | NO   |     | NULL    |       |
| phone_num   | varchar(10)  | YES  |     | NULL    |       |
| email       | varchar(50)  | NO   |     | NULL    |       |
| address     | varchar(100) | NO   |     | NULL    |       |
| idProof_num | int          | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| prop_id   | varchar(20) | NO   | PRI | NULL    |       |
| prop_mail | varchar(30) | NO   |     | NULL    |       |
| prop_name | char(20)    | NO   |     | NULL    |       |
| prop_type | char(12)    | NO   |     | NULL    |       |
| bhk       | int         | NO   |     | NULL    |       |
| prop_add  | varchar(50) | NO   |     | NULL    |       |
| prop_rent | int         | NO   |     | NULL    |       |
| status    | tinyint(1)  | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

```python
from tkinter import *
import mysql.connector
from tkinter import ttk
from datetime import datetime
import tkinter.messagebox as tkmessage
import smtplib
import random

con = mysql.connector.connect(
    host="localhost",
    user="root",
    password="tiger",
    database="dabbabnb"
)

cur = con.cursor(buffered=True)

def list_New_Property():
    root=Tk()
    root.title('ADD LISTING')
    root.geometry('720x720+0+0')
    Frame(root,bd=4,relief=RIDGE,bg='cyan').place(x=0,y=0,width=720,height=720 )
    Label(root,text='ADD NEW LISTING',bd=10,relief=GROOVE,font=('times new 
roman',40,'bold'),bg='yellow',fg='red').pack(side=TOP,fill=X)

    #==========================P_EMAIL====================================
    Label(root,text='Your E-Mail ID 
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=150)
    prop_mail=StringVar()
    prop_mail_entry=Entry(root,textvariable=prop_mail,width=25,bg='white').place(x=400,y=160)

    #==========================NAME====================================
    Label(root,text='Property Name 
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=230)
    prop_name=StringVar()
    prop_name_entry=Entry(root,textvariable=prop_name,width=25,bg='white').place(x=400,y=240)

    #==========================Prop type.================================
    Label(root,text='Type',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=310)
    prop_list=(' ','Appartment','Villa','Penthouse','Farmhouse')
    prop_type=StringVar()
    prop_type_entry=ttk.OptionMenu(root,prop_type,*prop_list).place(x=400,y=315)
    # Entry(root,textvariable=prop_type,width=25,bg='white').place(x=400,y=161)

    #==========================BHK====================================
    Label(root,text='No. of Bedrooms 
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=390)
    bhk_list=(' ','1','2','3','4','5','6','7','8','9','10')
    bhk=IntVar()
    bhk_entry=ttk.OptionMenu(root,bhk,*bhk_list).place(x=400,y=395)
    #=========================ADDRESS=====================================
    Label(root,text='Address
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=470)
    prop_address=StringVar()
    prop_address_entry=Entry(root,textvariable=prop_address,width=25,bg='white').place(x=400,
y=480)

    #=========================RENT=====================================
    Label(root,text='Rent ',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=550)
    prop_rent=IntVar()
    prop_rent_entry=Entry(root,textvariable=prop_rent,width=25,bg='white').place(x=400,y=560)

    def do_it():
        em=str(prop_mail.get()).upper()
        nm=str(prop_name.get()).upper()
        pt=str(prop_type.get()).upper()
        bh=int(bhk.get())
        add=str(prop_address.get()).upper()
        rt=int(prop_rent.get())
        st=1

        number=random.randint(100,999999)
        prop_id='P'+str(number)
        sender_email="landbnbn@outlook.com"
        password="landbnb@1234"
        TEXT='\n Please note your Property ID: '+prop_id
        SUBJECT='Property Listed Successfully.'
        message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

        server=smtplib.SMTP('smtp.office365.com',587)
        server.starttls()
        server.login(sender_email,password)
        server.sendmail(sender_email,em,message)


        cur.execute("insert into prop
values('{}','{}','{}','{}','{}','{}','{}','{}')".format(prop_id,em,nm,pt,bh,add,rt,st))
        con.commit()
        server=smtplib.SMTP('smtp.office365.com',587)
        server.starttls()
        server.login(sender_email,password)
        server.sendmail(sender_email,em,message)

        tkmessage.showinfo("Success!","Property Listed Successfully!")
        # prop_name_entry.delete(0, 'end')
        # prop_type_entry.delete(0, 'end')
        # bhk_entry.delete(0, 'end')
        # prop_address_entry.delete(0, 'end')
        # prop_rent_entry.delete(0, 'end')
        # print(' '*242+'Record Added Successfully...')


    Button(root,text='ADD',bd=10,relief=GROOVE,bg='lightblue',fg='navy blue',font=('times new
roman',40,'bold'),command=do_it).pack(side=BOTTOM, fill=X)

    root.mainloop()

def customer_button():      #customer details:- name, phone num, email, address, Id proof
    customer_details = Tk()
    customer_details.title('Property Details')
    customer_details.geometry('720x720+720+0')
    customer_details.iconbitmap('prop.ico')

    frame_for_heading = Frame(customer_details, bg="yellow", borderwidth=3, relief="raised")
    frame_for_heading.pack(fill=BOTH, anchor="c")
    Label(frame_for_heading, text="Booking Details", fg="blue", relief="ridge", font="goldman
19 bold").pack()

    #for checkin date
    Label(customer_details,text='DATE OF BIRTH
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=50)
    date_list=('
','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20',
'21','22','23','24','25','26','27','28','29','30','31')
    month_list=(' ','1','2','3','4','5','6','7','8','9','10','11','12')
    year_list=(' ','2022', '2023')
    date=StringVar()
    month=StringVar()
    year=StringVar()
    ttk.OptionMenu(customer_details, date, *date_list).place(x=400,y=50)
    ttk.OptionMenu(customer_details, month, *month_list).place(x=450,y=50)
    ttk.OptionMenu(customer_details, year, *year_list).place(x=540,y=50)

    #for cust name
    Label(customer_details,text='Name
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=130)
    cust_name=StringVar()
    cust_name_entry=Entry(customer_details,textvariable=cust_name,width=25,bg='white').place(
x=400,y=140)

    #for cust phone number
    Label(customer_details,text='Phone No.
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=230)
    cust_phone = IntVar()
    cust_phone_entry = Entry(customer_details, textvariable=cust_phone, width=25, bg =
'white').place(x=400, y= 240)

    #for email
    Label(customer_details,text='Email
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=330)
    cust_email = StringVar()
    cust_email_entry = Entry(customer_details, textvariable=cust_email, width=25, bg =
'white').place(x=400, y= 340)

    #for address
    Label(customer_details,text='Address
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=430)
    cust_Add = StringVar()
    cust_Add_Entry = Entry(customer_details, textvariable=cust_Add, width=25, bg =
'white').place(x=400, y= 440)

    #for idproof number
    Label(customer_details,text='ID proof number
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=530)
    cust_id = IntVar()
    cust_id_entry = Entry(customer_details, textvariable=cust_id, width=25, bg =
'white').place(x=400, y= 540)

    def do_it():
        dd = (date.get())
        mm = (month.get())
        yy = (year.get())
        namecust = str(cust_name.get())
        phonenum = int(cust_phone.get())
        email = str(cust_email.get())
        address = str(cust_Add.get())
        idNumber = int(cust_id.get())
        number=random.randint(100,999999)
        customer_id='C'+str(number)
        sqlq = f"insert into customer value('{customer_id}', '{namecust}', '{phonenum}',
'{email}', '{address}', {idNumber})"
        cur.execute(sqlq)
        con.commit()

    Button(customer_details,text='Confirm Book',bd=10,relief=GROOVE,bg='lightblue',fg='navy
blue',font=('times new roman',40,'bold'), command=do_it).pack(side=BOTTOM, fill=X)

    customer_details.mainloop()

def property_listing():
    pl = Tk()
    pl.geometry("720x720")
    pl.title("Property Listing")
    tree_frame = Frame(pl)
    tree_frame.pack(pady=10)
    tree_scrollbar=Scrollbar(tree_frame, )
    tree_scrollbar.pack(side="right", fill=Y)
    my_tree=ttk.Treeview(tree_frame, yscrollcommand=tree_scrollbar.set)  #, selectmode="none"
    my_tree.pack()

    #config of scrolbar
    tree_scrollbar.config(command=my_tree.yview)

    my_tree['columns'] = ("Pid", "Property name", "BHK", "Address", "Rent")

    my_tree.column("#0", width=0,stretch=NO)
    my_tree.column("Pid", anchor="w", width=60)
    my_tree.column("Property name", anchor="center", width=100)
    my_tree.column("BHK", anchor="w",width=60)
    my_tree.column("Address", anchor="w",width=150)
    my_tree.column("Rent", anchor="w",width=100)

    my_tree.heading("#0", text="", anchor="w")
    my_tree.heading("Pid", text="Pid", anchor="w")
    my_tree.heading("Property name",text="Name", anchor="center")
    my_tree.heading("BHK", text="BHK", anchor="w")
    my_tree.heading("Address", text="Address", anchor="w")
    my_tree.heading("Rent", text="Rent", anchor="w")

    fetchdata = "select prop_id, prop_name, BHK, prop_add, prop_rent from prop"
    cur.execute(fetchdata)
    con.commit()
    result_table_data = cur.fetchall()

    count = 0
    for rec in result_table_data:
        my_tree.insert(parent='', index='end', iid=count, text="", values=(rec[0], rec[1],
rec[2], rec[3], rec[4]))
        count += 1

    pl.mainloop()

def update_Listing():
    root=Tk()
    root.title('UPDATE LISTING')
    root.geometry('720x720+0+0')
    Frame(root,bd=4,relief=RIDGE,bg='cyan').place(x=0,y=0,width=720,height=720)
    Label(root,text='UPDATE LISTING ',bd=10,relief=GROOVE,font=('times new
roman',40,'bold'),bg='yellow',fg='red').pack(side=TOP,fill=X)

    #=========================PID=====================================
    Label(root,text='Property ID
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=150)
    prop_ID=StringVar()
    prop_ID_entry=Entry(root,textvariable=prop_ID,width=25,bg='white').place(x=400,y=160)

    #=========================NAME====================================
    Label(root,text='Property Name
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=230)
    prop_name=StringVar()
    prop_name_entry=Entry(root,textvariable=prop_name,width=25,bg='white').place(x=400,y=240)

    #===========================Prop type.=================================
    Label(root,text='Type',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=310)
    prop_list=(' ', 'NO CHANGE','Appartment','Villa','Penthouse','Farmhouse')
    prop_type=StringVar()
    prop_type_entry=ttk.OptionMenu(root,prop_type,*prop_list).place(x=400,y=315)
    # Entry(root,textvariable=prop_type,width=25,bg='white').place(x=400,y=161)

    #==========================BHK=====================================
    Label(root,text='No. of Bedrooms
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=390)
    bhk_list=(' ','0','1','2','3','4','5','6','7','8','9','10')
    bhk=IntVar()
    bhk_entry=ttk.OptionMenu(root,bhk,*bhk_list).place(x=400,y=395)

    #==========================ADDRESS====================================
    Label(root,text='Address
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=470)
    prop_address=StringVar()
    prop_address_entry=Entry(root,textvariable=prop_address,width=25,bg='white').place(x=400,
y=480)

    #==========================RENT====================================
    Label(root,text='Rent ',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=550)
    prop_rent=IntVar()
    prop_rent_entry=Entry(root,textvariable=prop_rent,width=25,bg='white').place(x=400,y=560)

    def do_it():                #WORK HERE
        id=str(prop_ID.get()).upper()
        nm=str(prop_name.get()).upper()
        pt=str(prop_type.get()).upper()
        bh=int(bhk.get())
        add=str(prop_address.get()).upper()
        rt=int(prop_rent.get())
        if nm != '0':
            cur.execute("update prop set prop_name=('{}') where prop_id=('{}')
".format(nm,id))
            con.commit()
            pass
        if pt != 'NO CHANGE':
            cur.execute("update prop set prop_type=('{}') where prop_id=('{}')
".format(pt,id))
            con.commit()
            pass
        if bh != '0':
            cur.execute("update prop set bhk=('{}') where prop_id=('{}') ".format(bh,id))
            con.commit()
            pass
        if add != '0':
            cur.execute("update prop set prop_add=('{}') where prop_id=('{}')
".format(add,id))
            con.commit()
            pass
        if rt != 0:
            cur.execute("update prop set prop_rent=('{}') where prop_id=('{}')
".format(rt,id))
            con.commit()
            pass

        print(' '*242+'Updation Successful...')

    Button(root,text='UPDATE',bd=10,relief=GROOVE,bg='lightblue',fg='navy blue',font=('times
new roman',40,'bold'),command=do_it).pack(side=BOTTOM, fill=X)

    root.mainloop()

def update_Booking():
    booking_window=Tk()
    booking_window.title('UPDATE BOOKING')
    booking_window.geometry('720x600+0+0')
    Frame(booking_window,bd=4,relief=RIDGE,bg='cyan').place(x=0,y=0,width=720,height=600)
    Label(booking_window,text='UPDATE BOOKING',bd=10,relief=GROOVE,font=('times new
roman',40,'bold'),bg='yellow',fg='red').pack(side=TOP,fill=X)

    #===========================BOOKING ID================================
    Label(booking_window,text='Booking ID
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=170)
    book_id=StringVar()
    book_id_entry=Entry(booking_window,textvariable=book_id,width=25,bg='white').place(x=400,
y=181)


    #=======================DOB========================================
    Label(booking_window,text='Date of
Booking',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=330)
    date_list=(' ','NO
CHANGE','1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19'
,'20','21','22','23','24','25','26','27','28','29','30','31')
    month_list=(' ','NO CHANGE','1','2','3','4','5','6','7','8','9','10','11','12')
    year_list=(' ','NO CHANGE','2022','2023')
    date=StringVar()
    month=StringVar()
    year=StringVar()
    OptionMenu(booking_window,date,*date_list).place(x=400,y=330)
    OptionMenu(booking_window,month,*month_list).place(x=480,y=330)
    OptionMenu(booking_window,year,*year_list).place(x=560,y=330)
    def do_it():
        an=str(book_id.get()).upper()
        db=str(date.get())+' '+str(month.get())+' '+str(year.get())

        if 'NO CHANGE' not in db :
            cur.execute("update booking set invoice_date =('{}') where book_id=('{}')
".format(db,an))
            con.commit()
        # print(' '*242+'Updation Successful...')

    Button(booking_window,text='UPDATE',bd=10,relief=GROOVE,bg='lightblue',fg='navy
blue',font=('times new roman',40,'bold'),command=do_it).pack(side=BOTTOM, fill=X)

    booking_window.mainloop()

def delete_listing():
    delete_listing_window=Tk()
    delete_listing_window.title('DELETE LISTING')
    delete_listing_window.geometry('500x500+0+0')
    Frame(delete_listing_window,bd=4,relief=RIDGE,bg='cyan').place(x=0,y=0,width=720,height=6
00)
    Label(delete_listing_window,text='DELETE LISTING',bd=10,relief=GROOVE,font=('times new
roman',40,'bold'),bg='yellow',fg='red').pack(side=TOP,fill=X)

    #===========================PROP ID=================================
    Label(delete_listing_window,text='Property ID
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=150)
    prop_id=StringVar()
    prop_id_entry=Entry(delete_listing_window,textvariable=prop_id,width=25,bg='white').place
(x=325,y=161)

    def do_it():
        pid=str(prop_id.get()).upper()
        # cur.execute("delete from prop where prop_id=('{}') ".format(pid))
        # con.commit()
        # print(' '*242+'Deletion Successful...')

    Button(delete_listing_window,text='DELETE',bd=10,relief=GROOVE,bg='lightblue',fg='navy
blue',font=('times new roman',40,'bold'),command=do_it).pack(side=BOTTOM, fill=X)

    delete_listing_window.mainloop()

def delete_booking():
    delete_booking_window=Tk()
    delete_booking_window.title('DELETE BOOKING')
    delete_booking_window.geometry('500x500+0+0')
    Frame(delete_booking_window,bd=4,relief=RIDGE,bg='cyan').place(x=0,y=0,width=720,height=6
00)
```

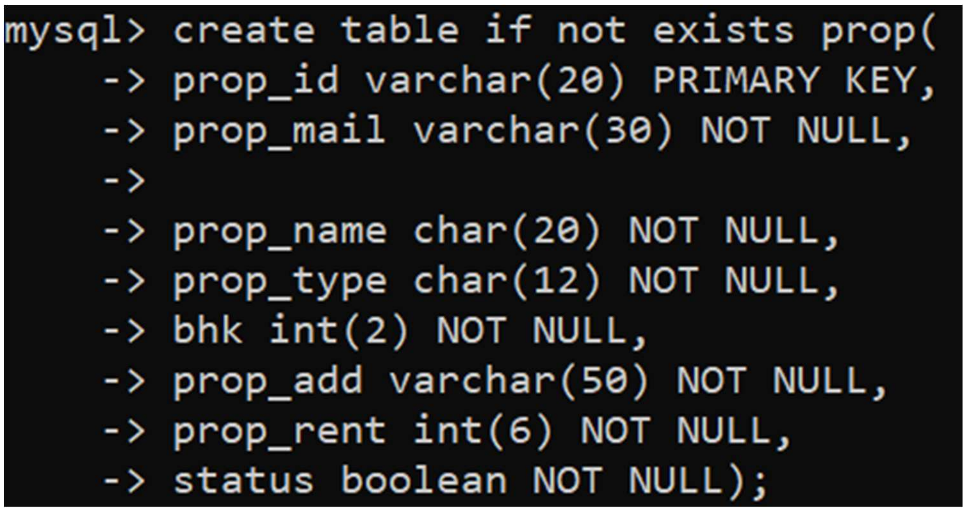
```
    Label(delete_booking_window,text='DELETE BOOKING',bd=10,relief=GROOVE,font=('times new
roman',40,'bold'),bg='yellow',fg='red').pack(side=TOP,fill=X)

    #===========================BOOKING ID.=================================
    Label(delete_booking_window,text='Booking ID
',bg='cyan',fg='black',font=('arial',20,'bold')).place(x=30,y=150)
    book_id=StringVar()
    book_id_entry=Entry(delete_booking_window,textvariable=book_id,width=25,bg='white').place
(x=325,y=161)

    def do_it():
        bid=str(book_id.get()).upper()
        # cur.execute("delete from booking where book_id=('{}') ".format(bid))
        # con.commit()
        # print(' '*242+'Deletion Successful...')

    Button(delete_booking_window,text='DELETE',bd=10,relief=GROOVE,bg='lightblue',fg='navy
blue',font=('times new roman',40,'bold'),command=do_it).pack(side=BOTTOM, fill=X)

    delete_booking_window.mainloop()

def aggregate():
    agg = Tk()
    agg.geometry("720x400+0+0")
    agg.title("Perform aggregate functions")
    agg.iconbitmap("prop.ico")

    def cheapest_prop():
        sql = "select MIN(prop_rent) from prop"
        cur.execute(sql)
        con.commit()
        result = cur.fetchone()
        tkmessage.showinfo("Cheapest property", f"The cheaptest property avaiable right now
is {result}")

    def available_prop():
        sql = "select COUNT(prop_id) from prop where status = 1"
        cur.execute(sql)
        con.commit()
        result = cur.fetchone()
        tkmessage.showinfo("Available property", f"The amount of total property available
right now is {result}" )

    def expensive_prop():
        sql = "select MAX(prop_rent) from prop"
        cur.execute(sql)
        con.commit()
        result = cur.fetchone()
        tkmessage.showinfo("Expensive property", f"The most expensive property available
right now is {result}")
```

```
    Label(agg, text="Perform aggregate function",
bg='cyan',fg='black',font=('arial',20,'bold')).pack(side=TOP, fill=X)
    butt1 = Button(agg, text='Cheapest Property', bd=10,relief=GROOVE,bg='lightblue',fg='navy
blue',font=('times new roman',40,'bold'), command=cheapest_prop)
    butt1.pack(side=TOP, fill=X)

    butt2 = Button(agg, text='Available property',
bd=10,relief=GROOVE,bg='lightblue',fg='navy blue',font=('times new roman',40,'bold'),
command=available_prop)
    butt2.pack(side=TOP, fill=X)

    butt3 = Button(agg, text='Most Expensive Property',
bd=10,relief=GROOVE,bg='lightblue',fg='navy blue',font=('times new roman',40,'bold'),
command=expensive_prop)
    butt3.pack(side=TOP, fill=X)

    agg.mainloop()

def KILLSWITCH():
    master.destroy()

if __name__ == "__main__":
    master = Tk()
    master.iconbitmap("prop.ico")

    master.geometry("500x500")
    master.minsize(300, 300)
    master.title("DabbaBnB")

    frame = Frame(master, bg="yellow", borderwidth=3, relief="raised")
    frame.pack(fill=BOTH, anchor="c")
    Label(frame, text="Welcome to DabbaBnB", fg="blue", relief="ridge", font="goldman 19
bold").pack()

    #button for listing a new property in the database
    frame_for_list = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_list.pack(anchor="center", pady=10)
    b1_list = Button(frame_for_list, text="List new property", command=list_New_Property)
    b1_list.pack(anchor="center")

    #button for booking a new property
    frame_for_book = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_book.pack(anchor="center", pady=10)
    b2_book = Button(frame_for_book, text="Do a booking",
command=lambda:[property_listing(),customer_button()]) #, command=remove
    b2_book.pack(anchor="center")

    #button for updating details about already listed property
    frame_for_updateL = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_updateL.pack(anchor="center", pady=10)
    b4_updateL = Button(frame_for_updateL, text="Update listing", command=update_Listing) #, 
command=show_bill
    b4_updateL.pack()

    #button for updating booking already booked
    frame_for_updateR = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_updateR.pack(anchor="center", pady=10)
    b3_updateR = Button(frame_for_updateR, text="Update Booking", command=update_Booking) #, 
command=LogOff
    b3_updateR.pack()

    #button for delete listing
    frame_for_deleteListing = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_deleteListing.pack(anchor="center", pady=10)
    b4_deleteListing = Button(frame_for_deleteListing, text="Delete Listing", 
command=delete_listing) #, command=LogOff
    b4_deleteListing.pack()

    #button for canceling a booking
    frame_for_deleteBooking = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_deleteBooking.pack(anchor="center", pady=10)
    b4_deleteBooking = Button(frame_for_deleteBooking, text="Cancel Booking", 
command=delete_booking) #, command=LogOff
    b4_deleteBooking.pack()

    frame_for_aggregate = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_aggregate.pack(anchor="center", pady=10)
    b4_aggregate = Button(frame_for_aggregate, text="Aggregate", command=aggregate) #, 
command=LogOff
    b4_aggregate.pack()

    frame_for_quit = Frame(master, borderwidth="6", bg="grey", relief="raised")
    frame_for_quit.pack(anchor="center", pady=10)
    b4_quit = Button(frame_for_quit, text="KILLSWITCH", command=KILLSWITCH) #, command=LogOff
    b4_quit.pack()

    master.mainloop()
```

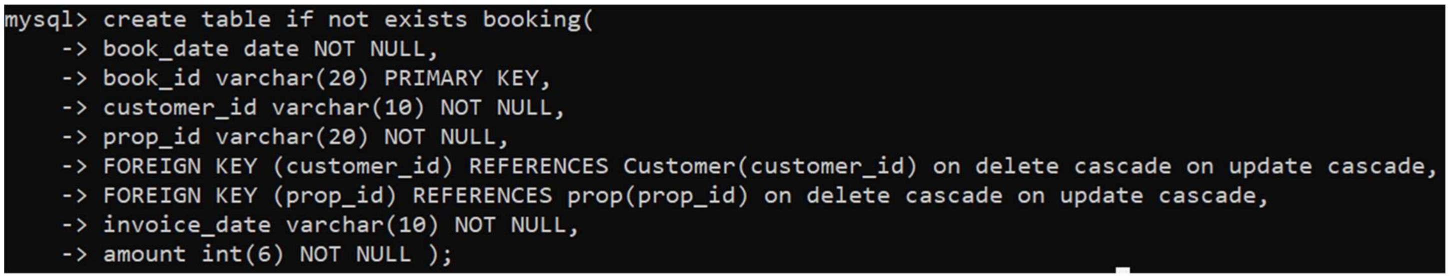
Create database and tables with proper constraints:



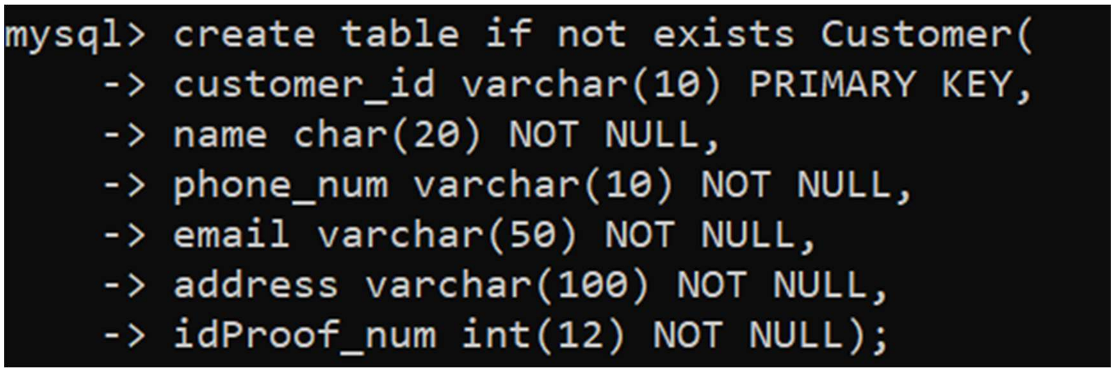
```
mysql> create database landbnb;
Query OK, 1 row affected (0.00 sec)
```



```
mysql> create table if not exists prop(
    -> prop_id varchar(20) PRIMARY KEY,
    -> prop_mail varchar(30) NOT NULL,
    ->
    -> prop_name char(20) NOT NULL,
    -> prop_type char(12) NOT NULL,
    -> bhk int(2) NOT NULL,
    -> prop_add varchar(50) NOT NULL,
    -> prop_rent int(6) NOT NULL,
    -> status boolean NOT NULL);
```

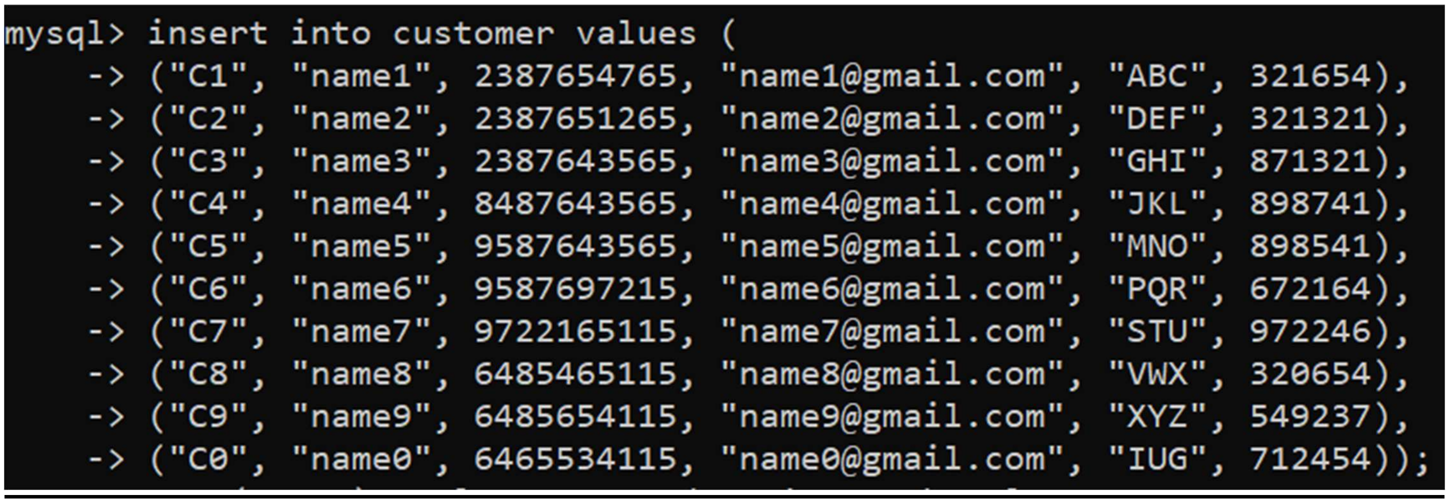


```
mysql> create table if not exists booking(
    -> book_date date NOT NULL,
    -> book_id varchar(20) PRIMARY KEY,
    -> customer_id varchar(10) NOT NULL,
    -> prop_id varchar(20) NOT NULL,
    -> FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) on delete cascade on update cascade,
    -> FOREIGN KEY (prop_id) REFERENCES prop(prop_id) on delete cascade on update cascade,
    -> invoice_date varchar(10) NOT NULL,
    -> amount int(6) NOT NULL );
```



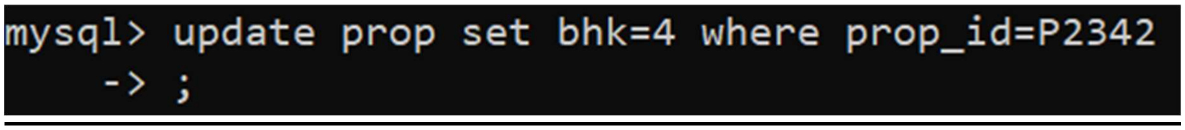
```
mysql> create table if not exists Customer(
    -> customer_id varchar(10) PRIMARY KEY,
    -> name char(20) NOT NULL,
    -> phone_num varchar(10) NOT NULL,
    -> email varchar(50) NOT NULL,
    -> address varchar(100) NOT NULL,
    -> idProof_num int(12) NOT NULL);
```

Inserting Values:



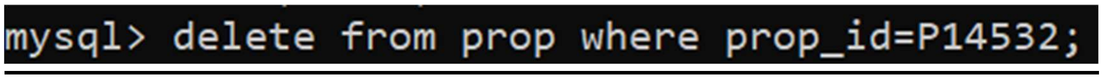
```
mysql> insert into customer values (
    -> ("C1", "name1", 2387654765, "name1@gmail.com", "ABC", 321654),
    -> ("C2", "name2", 2387651265, "name2@gmail.com", "DEF", 321321),
    -> ("C3", "name3", 2387643565, "name3@gmail.com", "GHI", 871321),
    -> ("C4", "name4", 8487643565, "name4@gmail.com", "JKL", 898741),
    -> ("C5", "name5", 9587643565, "name5@gmail.com", "MNO", 898541),
    -> ("C6", "name6", 9587697215, "name6@gmail.com", "PQR", 672164),
    -> ("C7", "name7", 9722165115, "name7@gmail.com", "STU", 972246),
    -> ("C8", "name8", 6485465115, "name8@gmail.com", "VWX", 320654),
    -> ("C9", "name9", 6485654115, "name9@gmail.com", "XYZ", 549237),
    -> ("C0", "name0", 6465534115, "name0@gmail.com", "IUG", 712454));
```

UPDATE RECORDS:



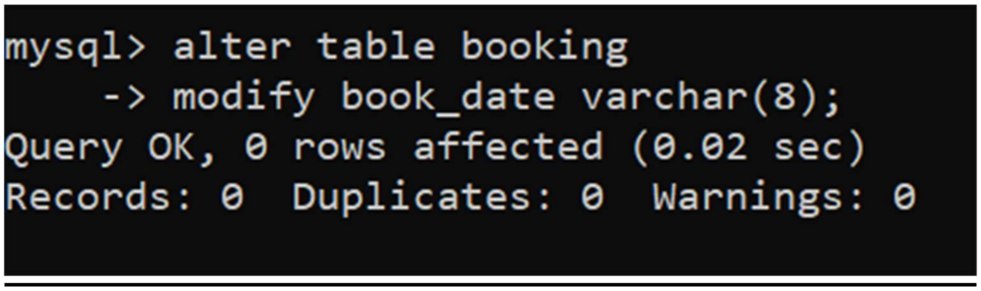
```
mysql> update prop set bhk=4 where prop_id=P2342
    -> ;
```

DELETE RECORDS:



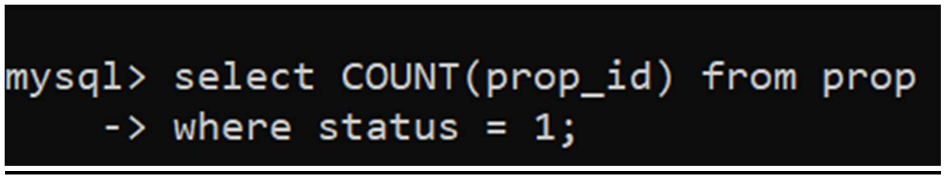
```
mysql> delete from prop where prop_id=P14532;
```

ALTER THE SCHEMA:



```
mysql> alter table booking
    -> modify book_date varchar(8);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Performing aggregate function:



```
mysql> select COUNT(prop_id) from prop
    -> where status = 1;
```