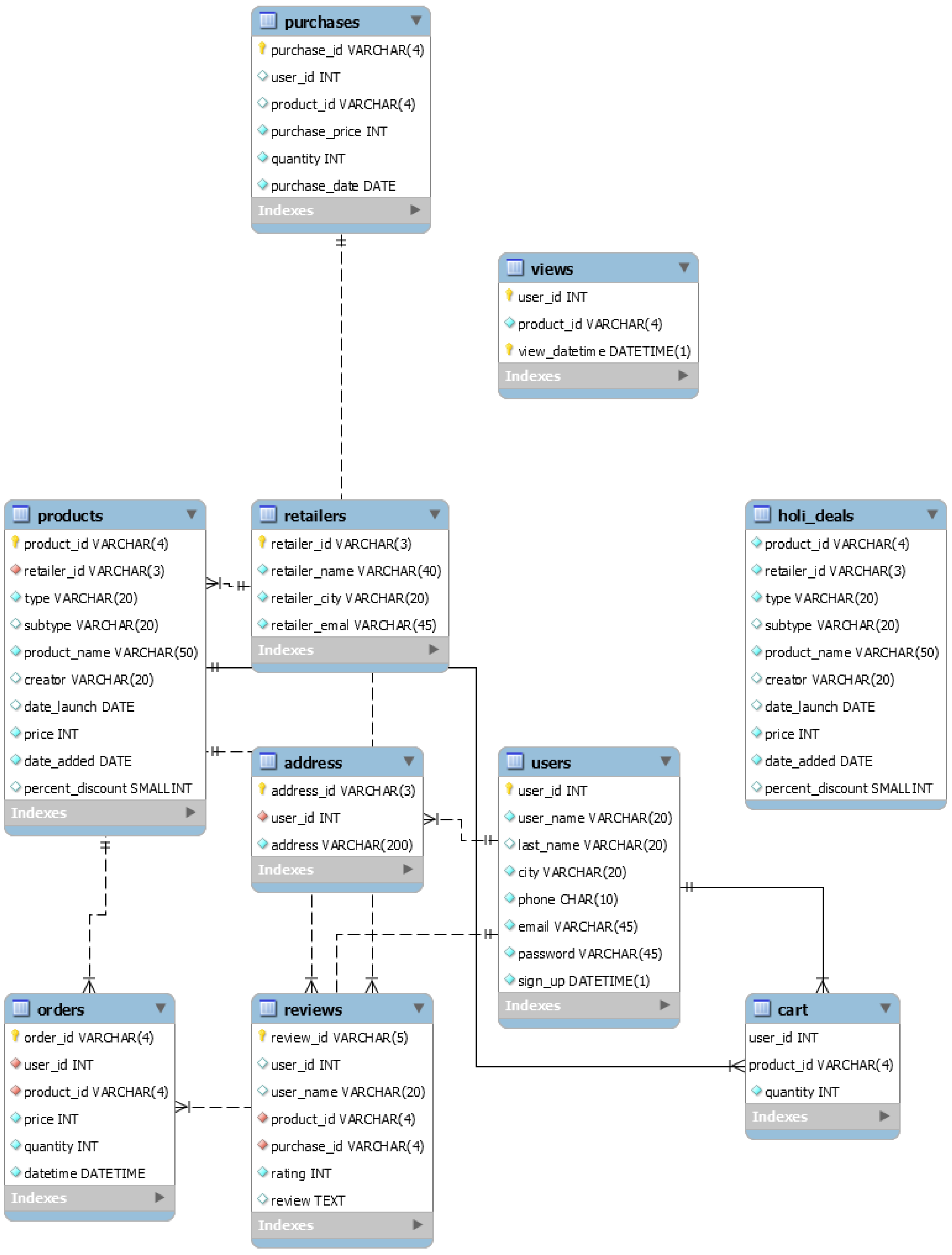
CS 432 Databases

Assignment - 4

Instructor : Prof. Mayank Singh

Devanshu Thakar
nilesh.thakar@iitgn.ac.in
18110174

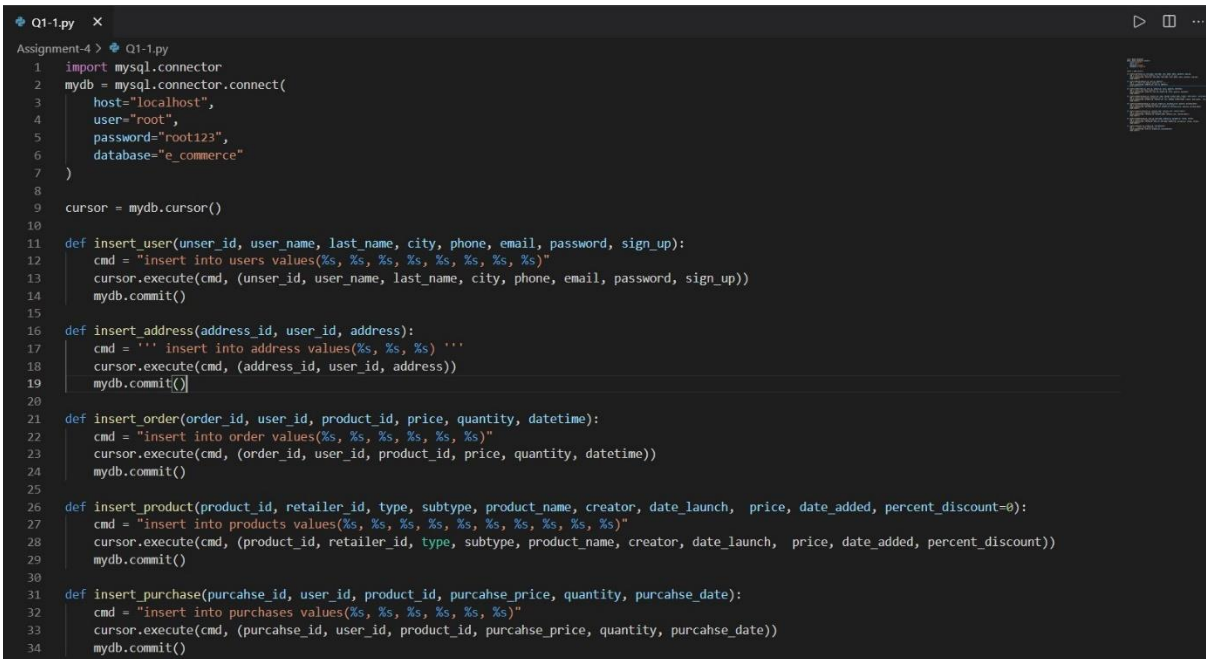
The schema Diagram for ecommerce_x database is shown below :



Q1

1. Method definition for filling the tables with at least 20 dummy records. (Please ensure that the database constraints are satisfied.)

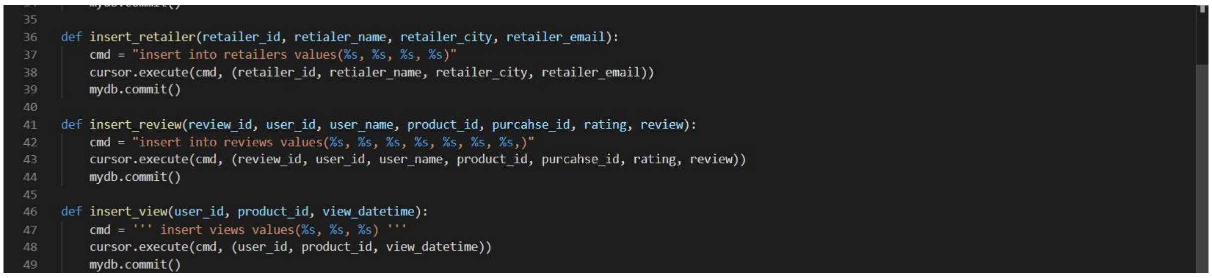
(Only function has been defined as the database was already having 20 records in every table)



Q1-1.py

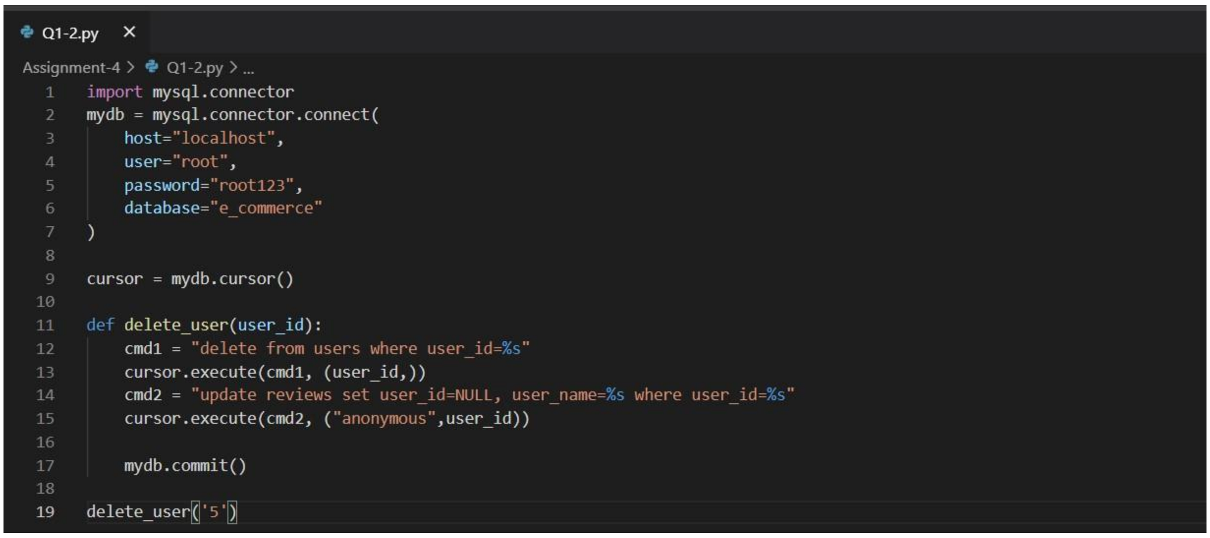
Assignment-4 > Q1-1.py

```python
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def insert_user(unser_id, user_name, last_name, city, phone, email, password, sign_up):
    cmd = "insert into users values(%s, %s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(cmd, (unser_id, user_name, last_name, city, phone, email, password, sign_up))
    mydb.commit()

def insert_address(address_id, user_id, address):
    cmd = ''' insert into address values(%s, %s, %s) '''
    cursor.execute(cmd, (address_id, user_id, address))
    mydb.commit()

def insert_order(order_id, user_id, product_id, price, quantity, datetime):
    cmd = "insert into order values(%s, %s, %s, %s, %s, %s)"
    cursor.execute(cmd, (order_id, user_id, product_id, price, quantity, datetime))
    mydb.commit()

def insert_product(product_id, retailer_id, type, subtype, product_name, creator, date_launch,  price, date_added, percent_discount=0):
    cmd = "insert into products values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(cmd, (product_id, retailer_id, type, subtype, product_name, creator, date_launch,  price, date_added, percent_discount))
    mydb.commit()

def insert_purchase(purcahse_id, user_id, product_id, purcahse_price, quantity, purcahse_date):
    cmd = "insert into purchases values(%s, %s, %s, %s, %s, %s)"
    cursor.execute(cmd, (purcahse_id, user_id, product_id, purcahse_price, quantity, purcahse_date))
    mydb.commit()
```



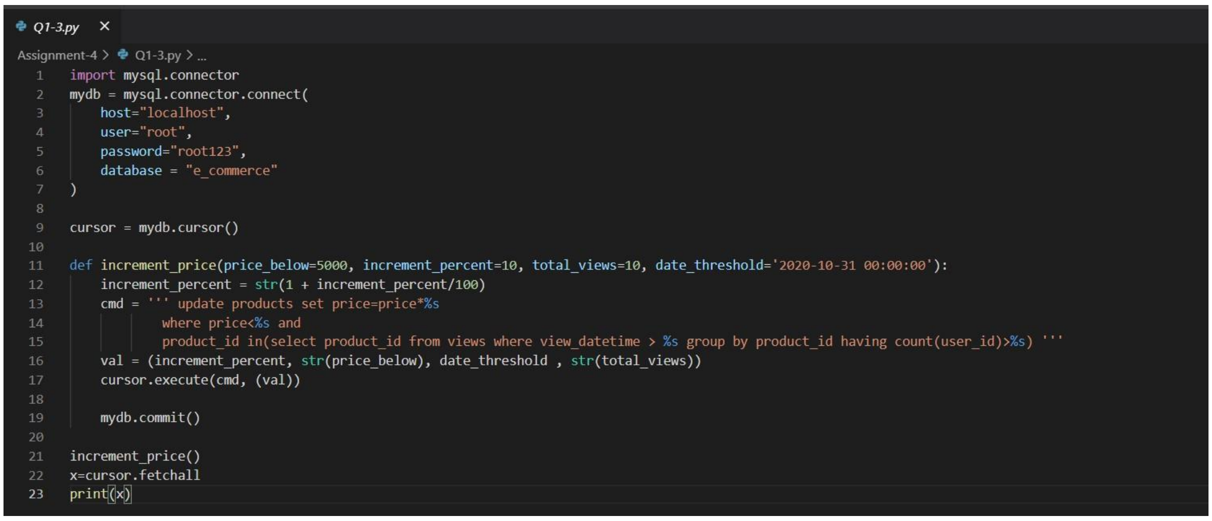
```python

def insert_retailer(retailer_id, retialer_name, retailer_city, retailer_email):
    cmd = "insert into retailers values(%s, %s, %s, %s)"
    cursor.execute(cmd, (retailer_id, retialer_name, retailer_city, retailer_email))
    mydb.commit()

def insert_review(review_id, user_id, user_name, product_id, purcahse_id, rating, review):
    cmd = "insert into reviews values(%s, %s, %s, %s, %s, %s, %s,)"
    cursor.execute(cmd, (review_id, user_id, user_name, product_id, purcahse_id, rating, review))
    mydb.commit()

def insert_view(user_id, product_id, view_datetime):
    cmd = ''' insert views values(%s, %s, %s) '''
    cursor.execute(cmd, (user_id, product_id, view_datetime))
    mydb.commit()
```

2. Delete a user from the database. After deleting the user update name of the user as ‘Anonymous’ in all the ratings and reviews written by that user.



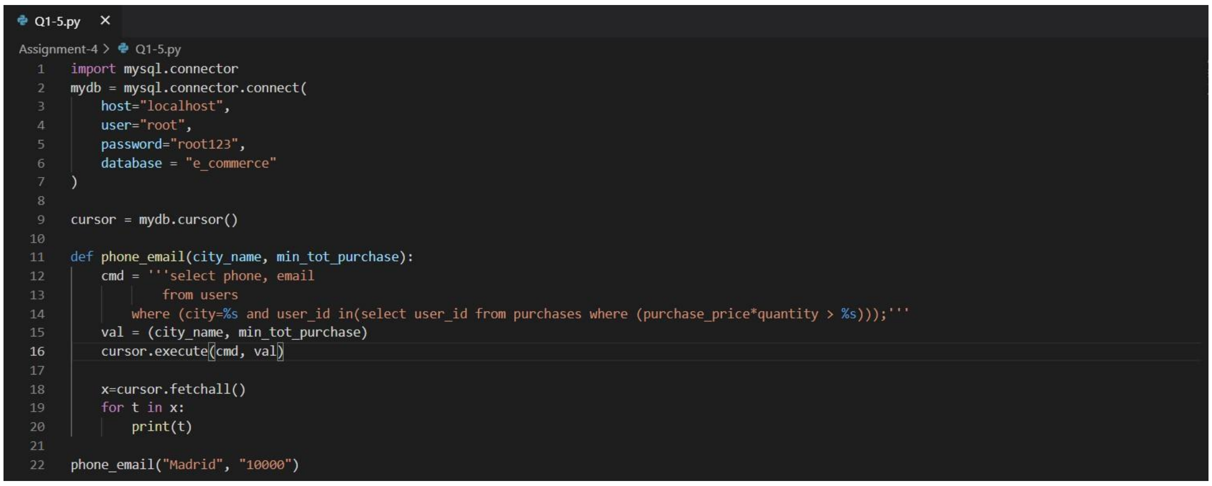
```python
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def delete_user(user_id):
    cmd1 = "delete from users where user_id=%s"
    cursor.execute(cmd1, (user_id,))
    cmd2 = "update reviews set user_id=NULL, user_name=%s where user_id=%s"
    cursor.execute(cmd2, ("anonymous",user_id))

    mydb.commit()

delete_user('5')
```

3. Increment the price of all products priced below Rs. 5000 by 10%, which were viewed by more than 10 users in the last 3 months.

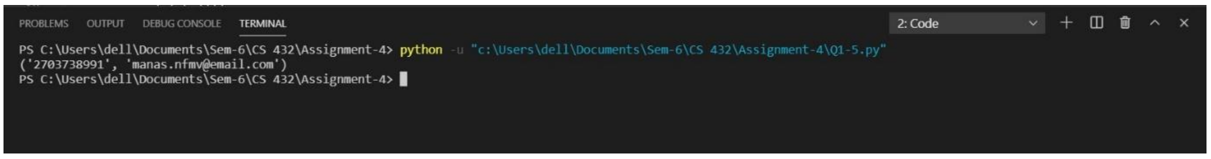


```python
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database = "e_commerce"
)

cursor = mydb.cursor()

def increment_price(price_below=5000, increment_percent=10, total_views=10, date_threshold='2020-10-31 00:00:00'):
    increment_percent = str(1 + increment_percent/100)
    cmd = ''' update products set price=price*%s
            where price<%s and
            product_id in(select product_id from views where view_datetime > %s group by product_id having count(user_id)>%s) '''
    val = (increment_percent, str(price_below), date_threshold , str(total_views))
    cursor.execute(cmd, (val))

    mydb.commit()

increment_price()
x=cursor.fetchall
print(x)
```

5. Find phone numbers and email IDs of all users who reside in the city ‘Madrid’ and have made a total purchase greater than or equal to Rs. 10000 in the past.

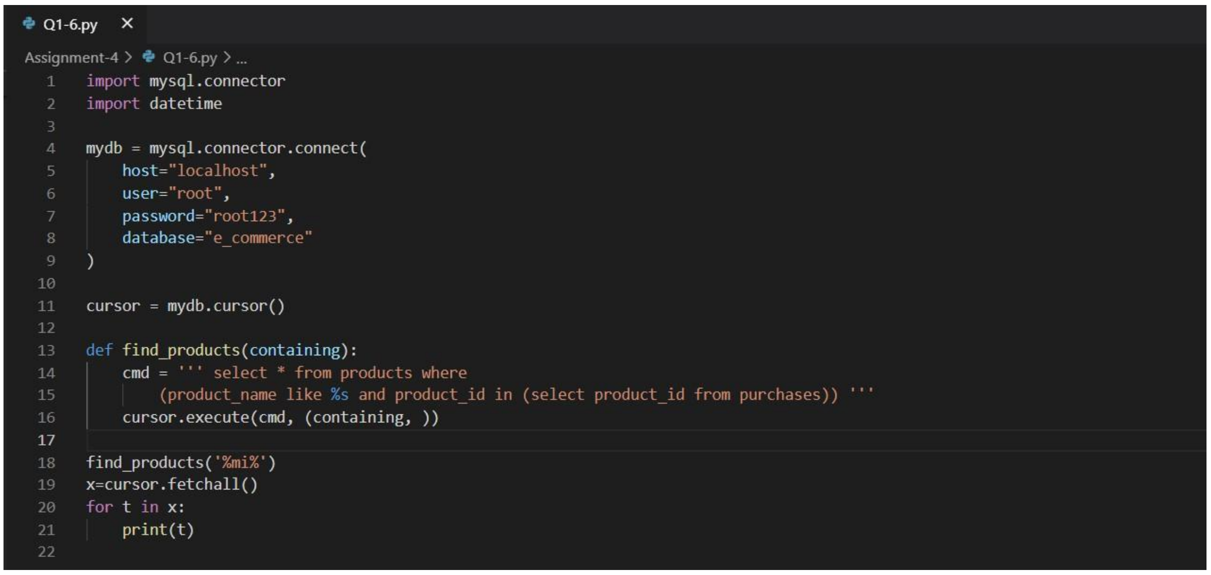


```python
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database = "e_commerce"
)

cursor = mydb.cursor()

def phone_email(city_name, min_tot_purchase):
    cmd = '''select phone, email
            from users
        where (city=%s and user_id in(select user_id from purchases where (purchase_price*quantity > %s)));'''
    val = (city_name, min_tot_purchase)
    cursor.execute(cmd, val)

    x=cursor.fetchall()
    for t in x:
        print(t)

phone_email("Madrid", "10000")
```

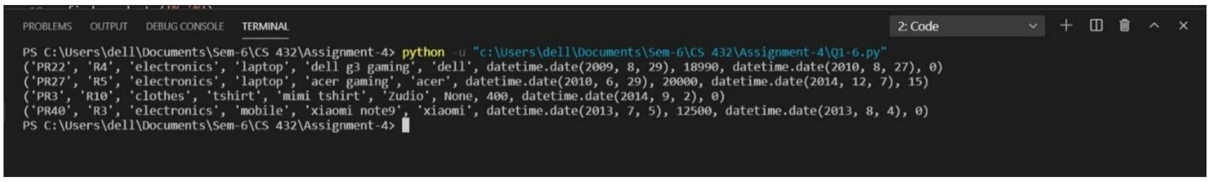


```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-5.py"
('2703738991', 'manas.nfmv@email.com')
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4>
```

6. Find all products in the database whose name contains the string ‘mi’. E.g. Xiaomi, etc, and all users who bought them at least once.

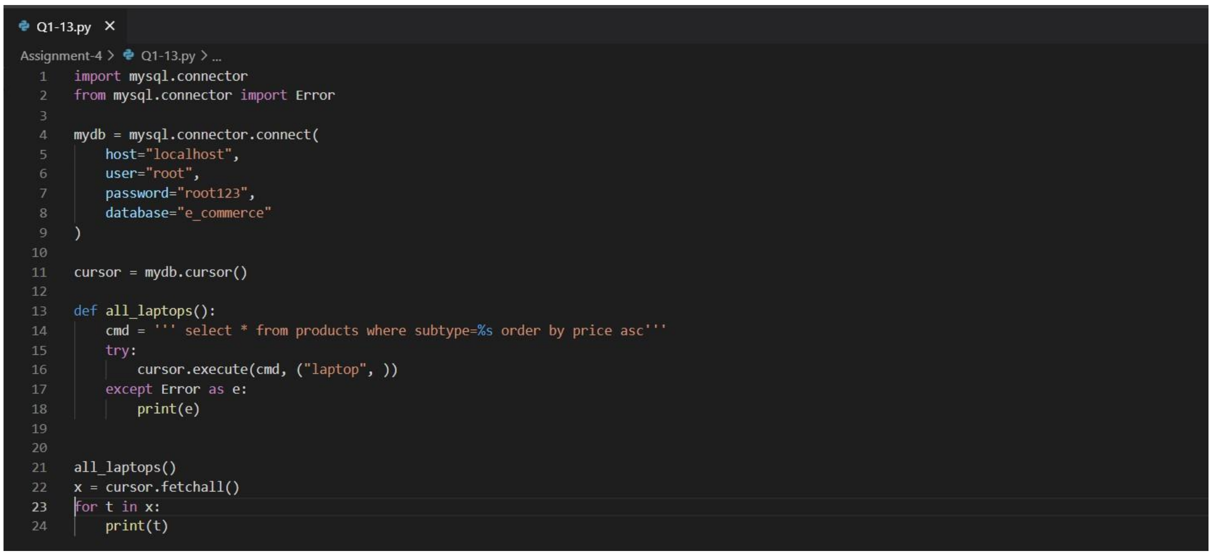


```python
import mysql.connector
import datetime

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def find_products(containing):
    cmd = ''' select * from products where
        (product_name like %s and product_id in (select product_id from purchases)) '''
    cursor.execute(cmd, (containing, ))

find_products('%mi%')
x=cursor.fetchall()
for t in x:
    print(t)
```

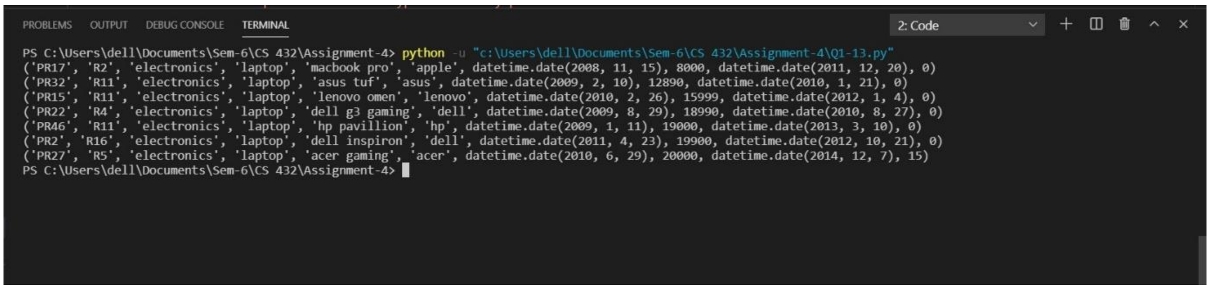


```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-6.py"
('PR22', 'R4', 'electronics', 'laptop', 'dell g3 gaming', 'dell', datetime.date(2009, 8, 29), 18990, datetime.date(2010, 8, 27), 0)
('PR27', 'R5', 'electronics', 'laptop', 'acer gaming', 'acer', datetime.date(2010, 6, 29), 20000, datetime.date(2014, 12, 7), 15)
('PR3', 'R10', 'clothes', 'tshirt', 'mimi tshirt', 'Zudio', None, 400, datetime.date(2014, 9, 2), 0)
('PR40', 'R3', 'electronics', 'mobile', 'xiaomi note9', 'xiaomi', datetime.date(2013, 7, 5), 12500, datetime.date(2013, 8, 4), 0)
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4>
```

13. Sort all laptops according to the price in increasing order.

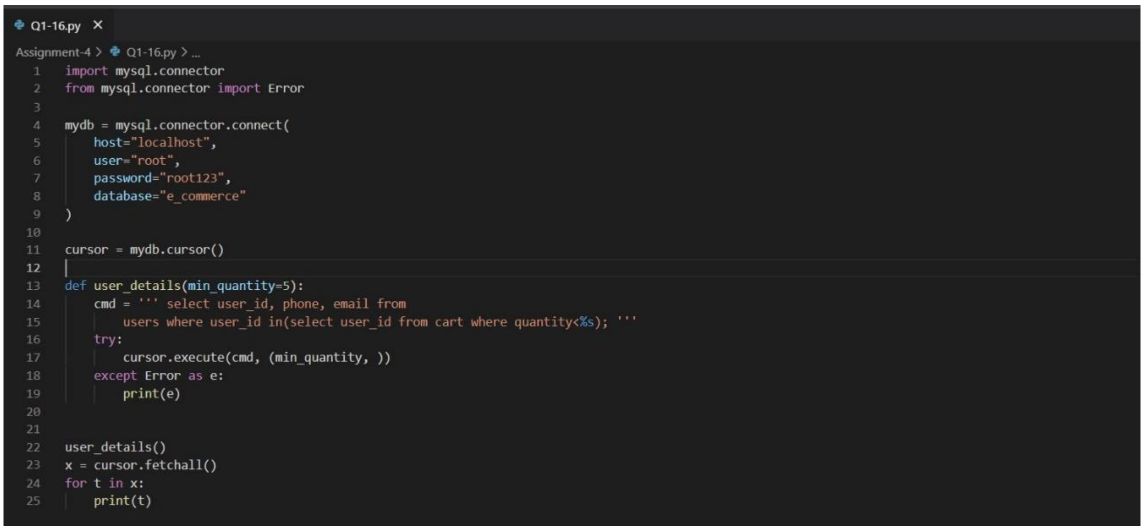


```python
import mysql.connector
from mysql.connector import Error

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def all_laptops():
    cmd = ''' select * from products where subtype=%s order by price asc'''
    try:
        cursor.execute(cmd, ("laptop", ))
    except Error as e:
        print(e)


all_laptops()
x = cursor.fetchall()
for t in x:
    print(t)
```

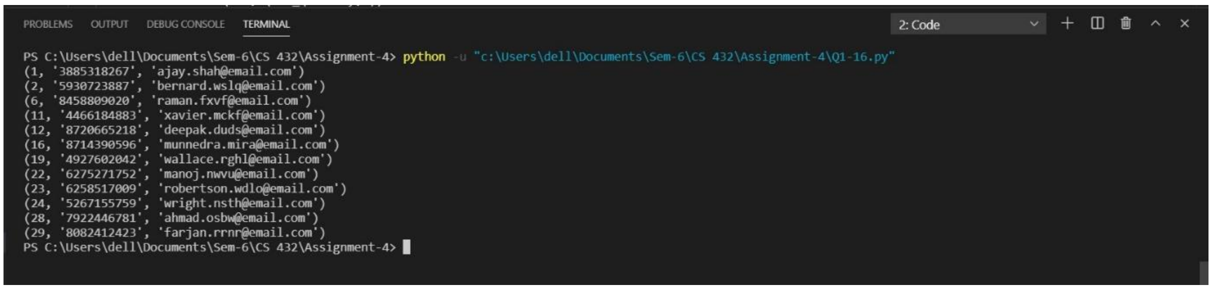


```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-13.py"
('PR17', 'R2', 'electronics', 'laptop', 'macbook pro', 'apple', datetime.date(2008, 11, 15), 8000, datetime.date(2011, 12, 20), 0)
('PR32', 'R11', 'electronics', 'laptop', 'asus tuf', 'asus', datetime.date(2009, 2, 10), 12890, datetime.date(2010, 1, 21), 0)
('PR15', 'R11', 'electronics', 'laptop', 'lenovo omen', 'lenovo', datetime.date(2010, 2, 26), 15999, datetime.date(2012, 1, 4), 0)
('PR22', 'R4', 'electronics', 'laptop', 'dell g3 gaming', 'dell', datetime.date(2009, 8, 29), 18990, datetime.date(2010, 8, 27), 0)
('PR46', 'R11', 'electronics', 'laptop', 'hp pavillion', 'hp', datetime.date(2009, 1, 11), 19000, datetime.date(2013, 3, 10), 0)
('PR2', 'R16', 'electronics', 'laptop', 'dell inspiron', 'dell', datetime.date(2011, 4, 23), 19900, datetime.date(2012, 10, 21), 0)
('PR27', 'R5', 'electronics', 'laptop', 'acer gaming', 'acer', datetime.date(2010, 6, 29), 20000, datetime.date(2014, 12, 7), 15)
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4>
```

16. Print the UserId, mobile number, and Email Id of all users who have saved a product in the cart, whose quantity is less than 5.

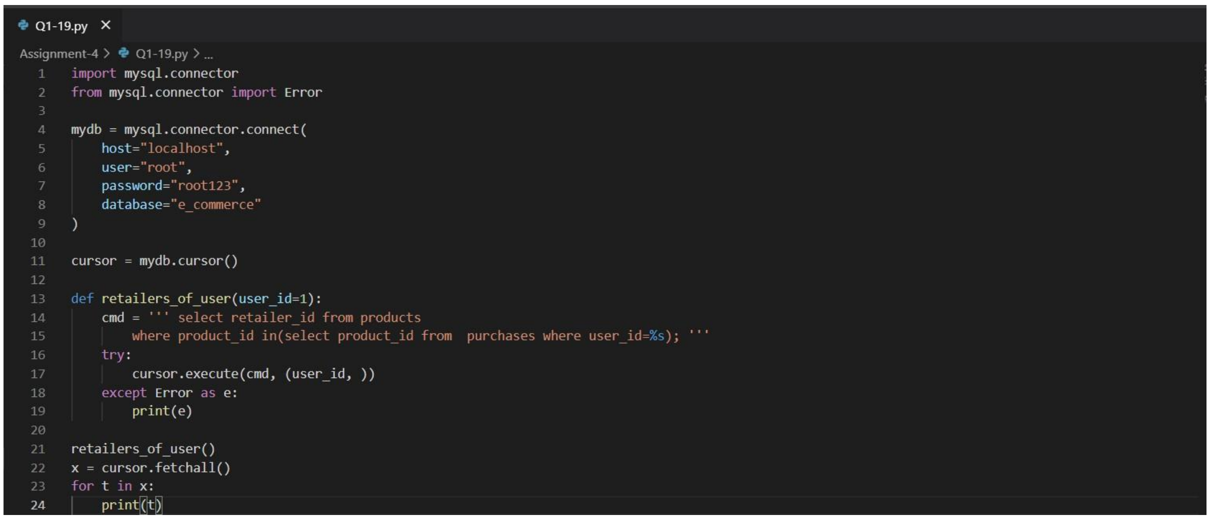


```python
import mysql.connector
from mysql.connector import Error

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def user_details(min_quantity=5):
    cmd = ''' select user_id, phone, email from
        users where user_id in(select user_id from cart where quantity<%s); '''
    try:
        cursor.execute(cmd, (min_quantity, ))
    except Error as e:
        print(e)


user_details()
x = cursor.fetchall()
for t in x:
    print(t)
```

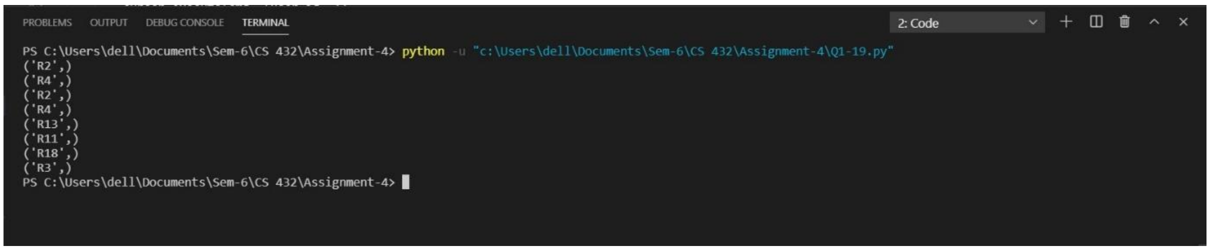


```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-16.py"
(1, '3885318267', 'ajay.shah@email.com')
(2, '5930723887', 'bernard.wslq@email.com')
(6, '8458809020', 'raman.fxvf@email.com')
(11, '4466184883', 'xavier.mckf@email.com')
(12, '8720665218', 'deepak.duds@email.com')
(16, '8714390596', 'munnedra.mira@email.com')
(19, '4927602042', 'wallace.rghl@email.com')
(22, '6275271752', 'manoj.nwvu@email.com')
(23, '6258517009', 'robertson.wdlo@email.com')
(24, '5267155759', 'wright.nsth@email.com')
(28, '7922446781', 'ahmad.osbw@email.com')
(29, '8082412423', 'farjan.rrnr@email.com')
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4>
```

19. List all retailer ids whose products, user_id = 1 have purchased.

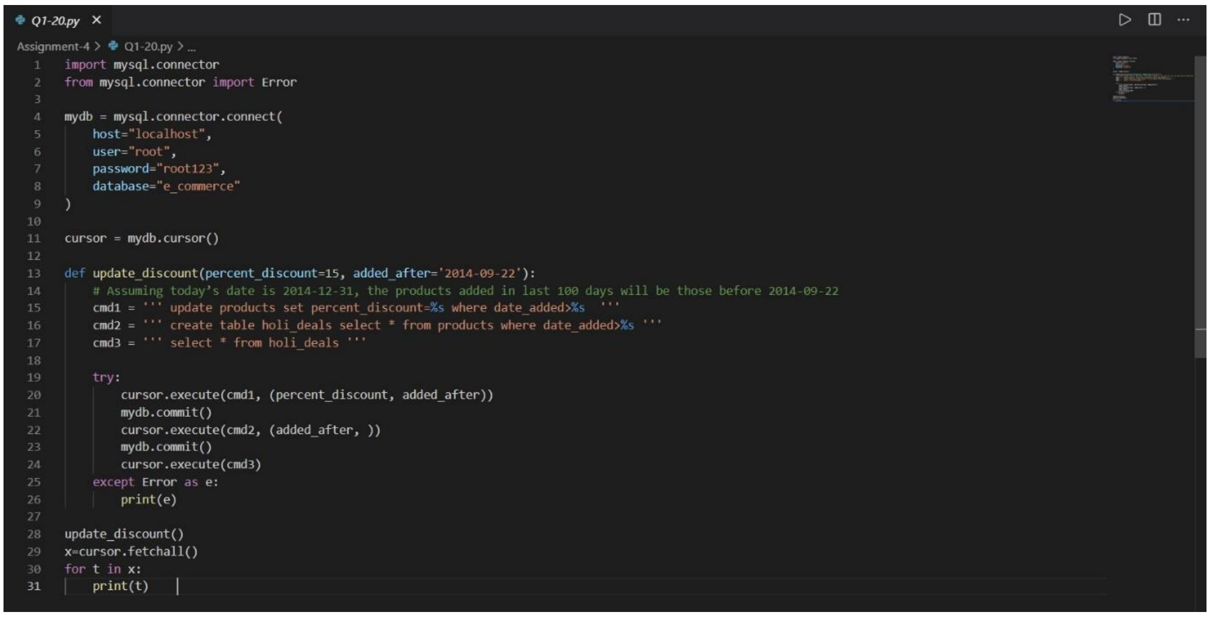


```python
import mysql.connector
from mysql.connector import Error

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def retailers_of_user(user_id=1):
    cmd = ''' select retailer_id from products
        where product_id in(select product_id from  purchases where user_id=%s); '''
    try:
        cursor.execute(cmd, (user_id, ))
    except Error as e:
        print(e)

retailers_of_user()
x = cursor.fetchall()
for t in x:
    print(t)
```

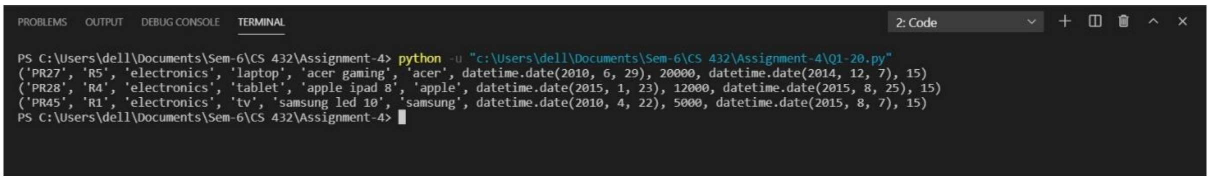


```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-19.py"
('R2',)
('R4',)
('R2',)
('R4',)
('R13',)
('R11',)
('R18',)
('R3',)
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> 
```

20. Write a query to update the discount on all new products by 15% and store it as a new table holi_Deals.



```python
import mysql.connector
from mysql.connector import Error

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def update_discount(percent_discount=15, added_after='2014-09-22'):
    # Assuming today's date is 2014-12-31, the products added in last 100 days will be those before 2014-09-22
    cmd1 = ''' update products set percent_discount=%s where date_added>%s  '''
    cmd2 = ''' create table holi_deals select * from products where date_added>%s '''
    cmd3 = ''' select * from holi_deals '''

    try:
        cursor.execute(cmd1, (percent_discount, added_after))
        mydb.commit()
        cursor.execute(cmd2, (added_after, ))
        mydb.commit()
        cursor.execute(cmd3)
    except Error as e:
        print(e)

update_discount()
x=cursor.fetchall()
for t in x:
    print(t)
```

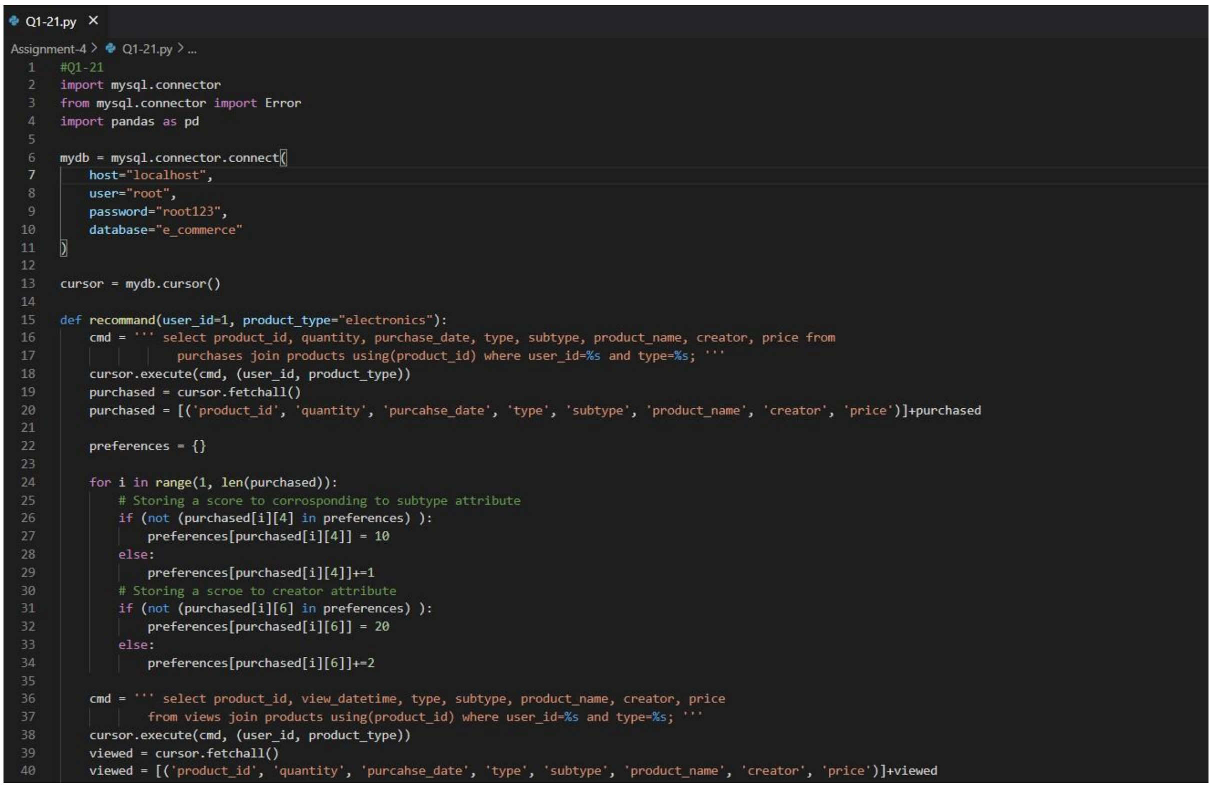


```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-20.py"
('PR27', 'R5', 'electronics', 'laptop', 'acer gaming', 'acer', datetime.date(2010, 6, 29), 20000, datetime.date(2014, 12, 7), 15)
('PR28', 'R4', 'electronics', 'tablet', 'apple ipad 8', 'apple', datetime.date(2015, 1, 23), 12000, datetime.date(2015, 8, 25), 15)
('PR45', 'R1', 'electronics', 'tv', 'samsung led 10', 'samsung', datetime.date(2010, 4, 22), 5000, datetime.date(2015, 8, 7), 15)
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> 
```

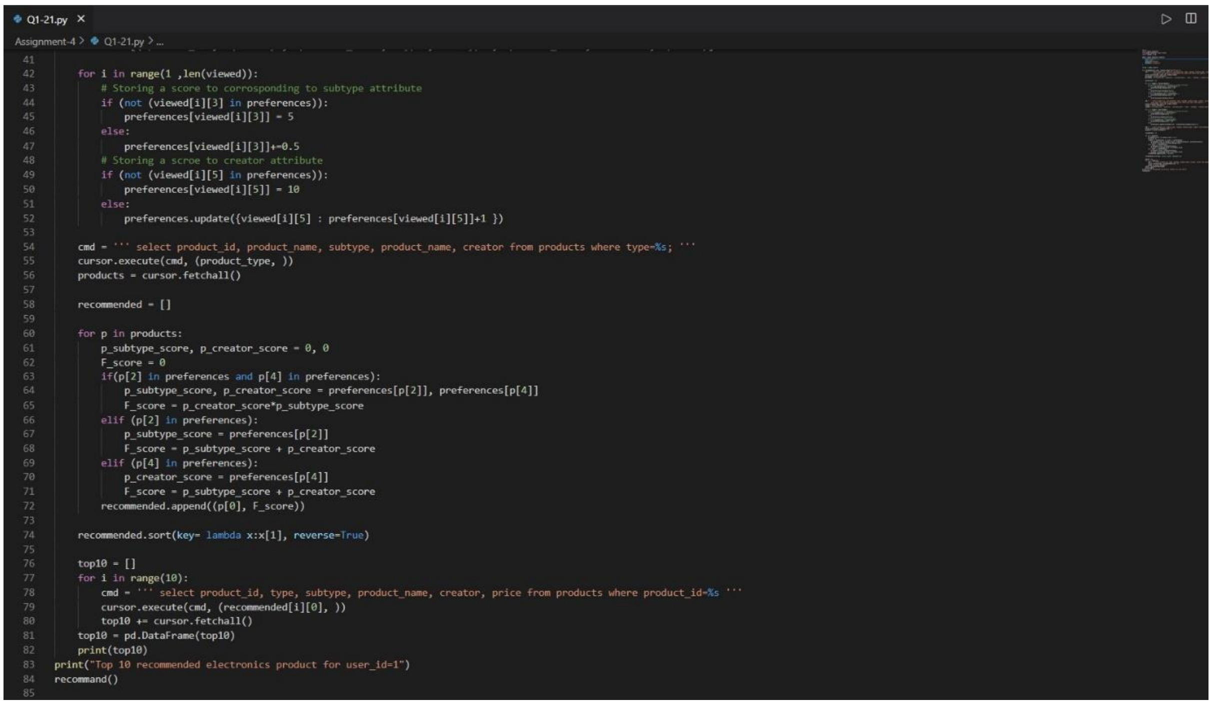
Note: Any product that is added to the database in the last 100 days is considered to be a new product.

21. List the top 10 recommended products for the user_id=1 based on the user's purchase and search history(use any recommendation algorithm).

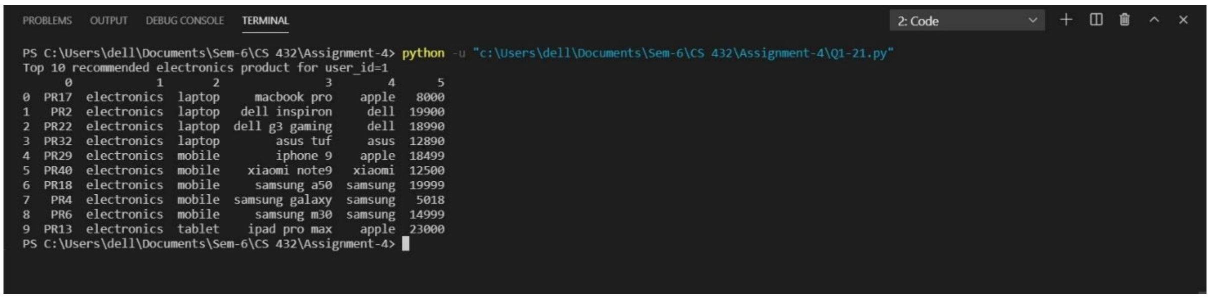
For recommendation a content based recommendation method was implemented. Two attributes, creator and subtype, from the *purchased* and *views* relation were used to understand the preference of the user. Creator in the belonging to the purchased table was initialized a score of 20 and subtype belonging to purchased table was initialized a score of 10. Similarly, creator in the views table was initialized a score of 10 and subtype with a score of 5. The mapping of attribute with score was done using python dictionary. The score a attribute was increased in every repetition of the attribute. Finally based on the preference dictionary, a score for each electronic product was computed and results were displayed in sorted order based on score.



```
#Q1-21
import mysql.connector
from mysql.connector import Error
import pandas as pd

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root123",
    database="e_commerce"
)

cursor = mydb.cursor()

def recommand(user_id=1, product_type="electronics"):
    cmd = ''' select product_id, quantity, purchase_date, type, subtype, product_name, creator, price from
                purchases join products using(product_id) where user_id=%s and type=%s; '''
    cursor.execute(cmd, (user_id, product_type))
    purchased = cursor.fetchall()
    purchased = [('product_id', 'quantity', 'purcahse_date', 'type', 'subtype', 'product_name', 'creator', 'price')]+purchased

    preferences = {}

    for i in range(1, len(purchased)):
        # Storing a score to corrosponding to subtype attribute
        if (not (purchased[i][4] in preferences) ):
            preferences[purchased[i][4]] = 10
        else:
            preferences[purchased[i][4]]+=1
        # Storing a scroe to creator attribute
        if (not (purchased[i][6] in preferences) ):
            preferences[purchased[i][6]] = 20
        else:
            preferences[purchased[i][6]]+=2

    cmd = ''' select product_id, view_datetime, type, subtype, product_name, creator, price
            from views join products using(product_id) where user_id=%s and type=%s; '''
    cursor.execute(cmd, (user_id, product_type))
    viewed = cursor.fetchall()
    viewed = [('product_id', 'quantity', 'purcahse_date', 'type', 'subtype', 'product_name', 'creator', 'price')]+viewed
```

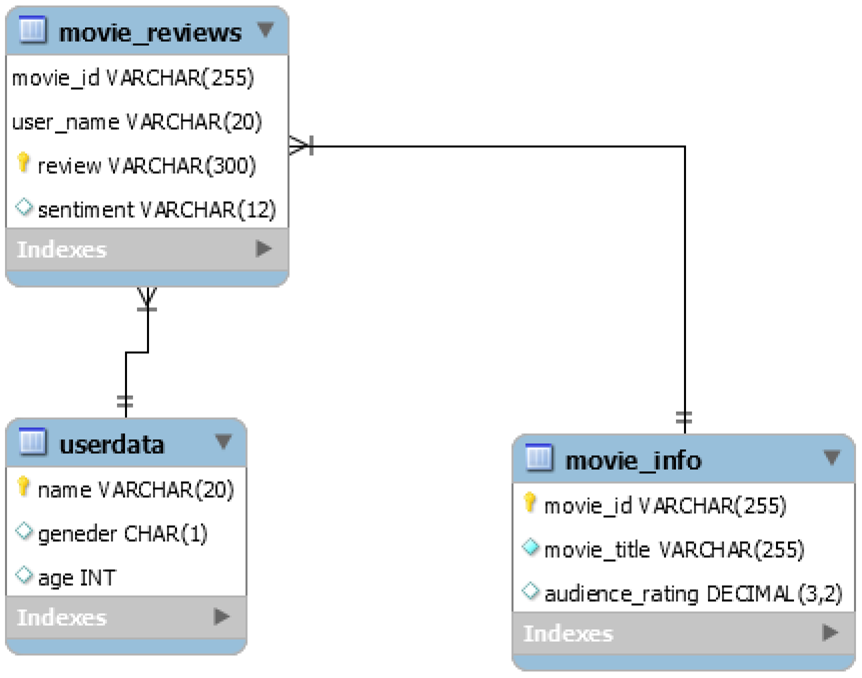


```
    for i in range(1 ,len(viewed)):
        # Storing a score to corrosponding to subtype attribute
        if (not (viewed[i][3] in preferences)):
            preferences[viewed[i][3]] = 5
        else:
            preferences[viewed[i][3]]+=0.5
        # Storing a scroe to creator attribute
        if (not (viewed[i][5] in preferences)):
            preferences[viewed[i][5]] = 10
        else:
            preferences.update({viewed[i][5] : preferences[viewed[i][5]]+1 })

    cmd = ''' select product_id, product_name, subtype, product_name, creator from products where type=%s; '''
    cursor.execute(cmd, (product_type, ))
    products = cursor.fetchall()

    recommended = []

    for p in products:
        p_subtype_score, p_creator_score = 0, 0
        F_score = 0
        if(p[2] in preferences and p[4] in preferences):
            p_subtype_score, p_creator_score = preferences[p[2]], preferences[p[4]]
            F_score = p_creator_score*p_subtype_score
        elif (p[2] in preferences):
            p_subtype_score = preferences[p[2]]
            F_score = p_subtype_score + p_creator_score
        elif (p[4] in preferences):
            p_creator_score = preferences[p[4]]
            F_score = p_subtype_score + p_creator_score
        recommended.append((p[0], F_score))

    recommended.sort(key= lambda x:x[1], reverse=True)

    top10 = []
    for i in range(10):
        cmd = ''' select product_id, type, subtype, product_name, creator, price from products where product_id=%s '''
        cursor.execute(cmd, (recommended[i][0], ))
        top10 += cursor.fetchall()
    top10 = pd.DataFrame(top10)
    print(top10)
print("Top 10 recommended electronics product for user_id=1")
recommand()
```



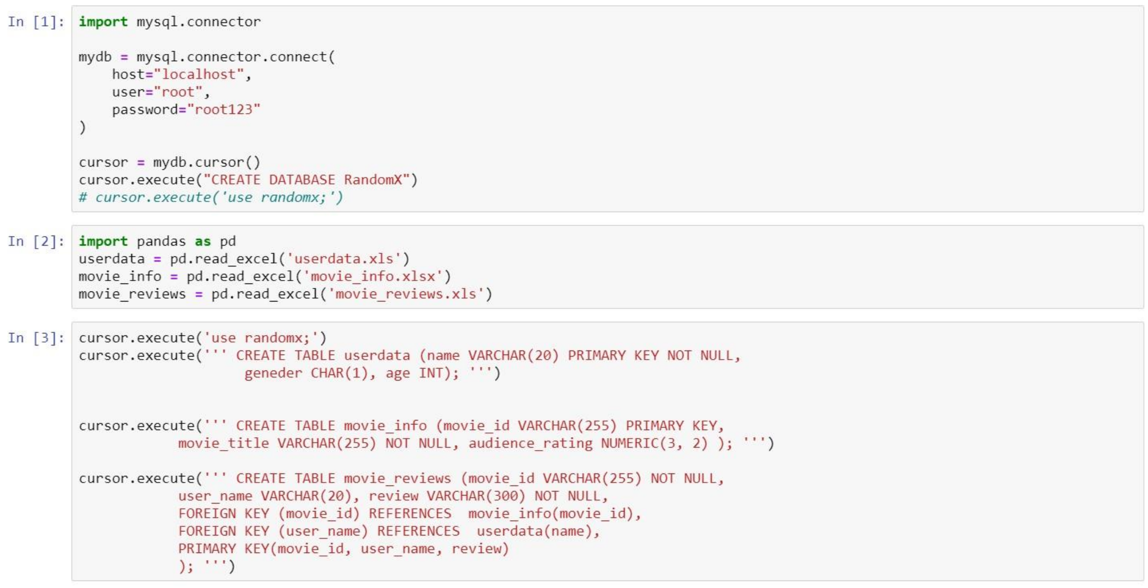
```
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> python -u "c:\Users\dell\Documents\Sem-6\CS 432\Assignment-4\Q1-21.py"
Top 10 recommended electronics product for user_id=1
      0            1       2               3        4      5
0  PR17  electronics  laptop     macbook pro    apple   8000
1   PR2  electronics  laptop   dell inspiron     dell  19900
2  PR22  electronics  laptop  dell g3 gaming     dell  18990
3  PR32  electronics  laptop        asus tuf     asus  12890
4  PR29  electronics  mobile        iphone 9    apple  18499
5  PR40  electronics  mobile    xiaomi note9   xiaomi  12500
6  PR18  electronics  mobile     samsung a50  samsung  19999
7   PR4  electronics  mobile  samsung galaxy  samsung   5018
8   PR6  electronics  mobile     samsung m30  samsung  14999
9  PR13  electronics  tablet    ipad pro max    apple  23000
PS C:\Users\dell\Documents\Sem-6\CS 432\Assignment-4> 
```

The Schema Diagram for random_X is shown below :



Q2

a.

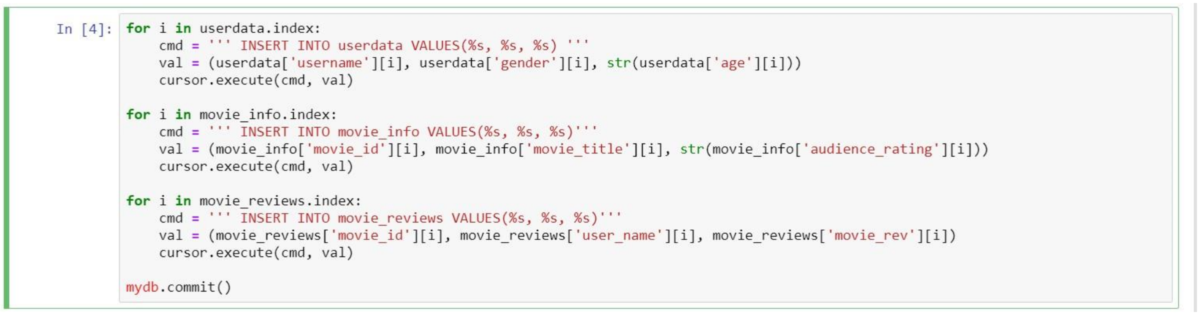


```
In [1]: import mysql.connector

        mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root123"
        )

        cursor = mydb.cursor()
        cursor.execute("CREATE DATABASE RandomX")
        # cursor.execute('use randomx;')
```

```
In [2]: import pandas as pd
        userdata = pd.read_excel('userdata.xls')
        movie_info = pd.read_excel('movie_info.xlsx')
        movie_reviews = pd.read_excel('movie_reviews.xls')
```

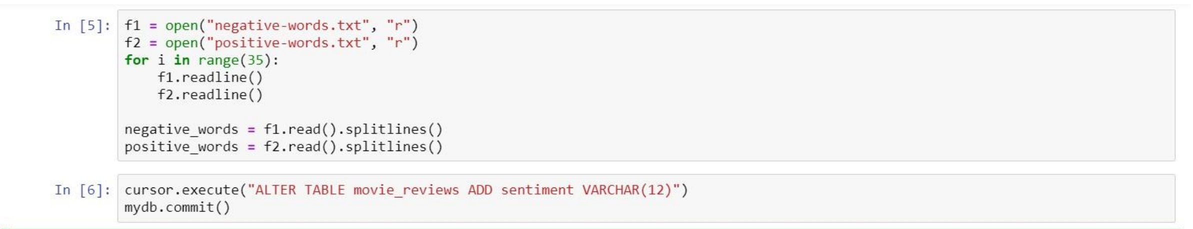
```
In [3]: cursor.execute('use randomx;')
        cursor.execute(''' CREATE TABLE userdata (name VARCHAR(20) PRIMARY KEY NOT NULL,
                            geneder CHAR(1), age INT); ''')


        cursor.execute(''' CREATE TABLE movie_info (movie_id VARCHAR(255) PRIMARY KEY,
                    movie_title VARCHAR(255) NOT NULL, audience_rating NUMERIC(3, 2) ); ''')

        cursor.execute(''' CREATE TABLE movie_reviews (movie_id VARCHAR(255) NOT NULL,
                    user_name VARCHAR(20), review VARCHAR(300) NOT NULL,
                    FOREIGN KEY (movie_id) REFERENCES  movie_info(movie_id),
                    FOREIGN KEY (user_name) REFERENCES  userdata(name),
                    PRIMARY KEY(movie_id, user_name, review)
                    ); ''')
```

b.

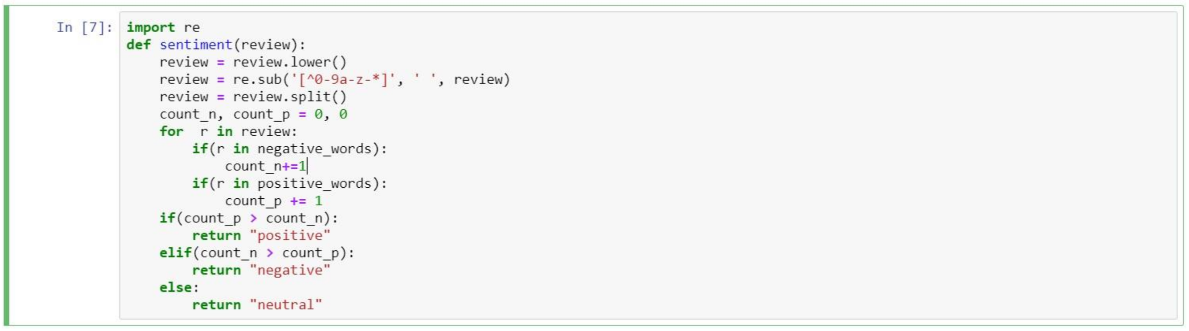


```
In [4]: for i in userdata.index:
            cmd = ''' INSERT INTO userdata VALUES(%s, %s, %s) '''
            val = (userdata['username'][i], userdata['gender'][i], str(userdata['age'][i]))
            cursor.execute(cmd, val)

        for i in movie_info.index:
            cmd = ''' INSERT INTO movie_info VALUES(%s, %s, %s)'''
            val = (movie_info['movie_id'][i], movie_info['movie_title'][i], str(movie_info['audience_rating'][i]))
            cursor.execute(cmd, val)

        for i in movie_reviews.index:
            cmd = ''' INSERT INTO movie_reviews VALUES(%s, %s, %s)'''
            val = (movie_reviews['movie_id'][i], movie_reviews['user_name'][i], movie_reviews['movie_rev'][i])
            cursor.execute(cmd, val)

        mydb.commit()
```

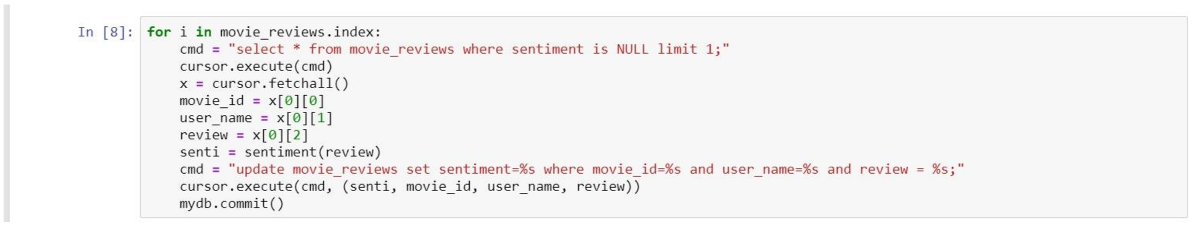
c.



```
In [5]: f1 = open("negative-words.txt", "r")
        f2 = open("positive-words.txt", "r")
        for i in range(35):
            f1.readline()
            f2.readline()

        negative_words = f1.read().splitlines()
        positive_words = f2.read().splitlines()

In [6]: cursor.execute("ALTER TABLE movie_reviews ADD sentiment VARCHAR(12)")
        mydb.commit()
```

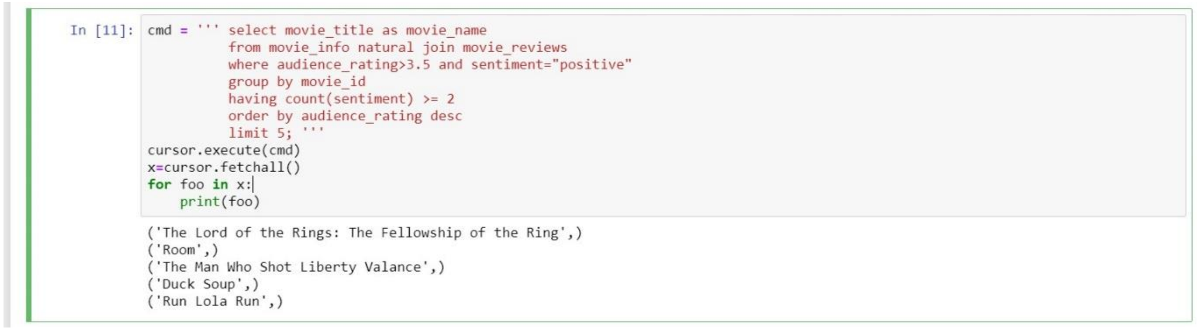


```
In [7]: import re
        def sentiment(review):
            review = review.lower()
            review = re.sub('[^0-9a-z-*]', ' ', review)
            review = review.split()
            count_n, count_p = 0, 0
            for  r in review:
                if(r in negative_words):
                    count_n+=1
                if(r in positive_words):
                    count_p += 1
            if(count_p > count_n):
                return "positive"
            elif(count_n > count_p):
                return "negative"
            else:
                return "neutral"
```



```
In [8]: for i in movie_reviews.index:
            cmd = "select * from movie_reviews where sentiment is NULL limit 1;"
            cursor.execute(cmd)
            x = cursor.fetchall()
            movie_id = x[0][0]
            user_name = x[0][1]
            review = x[0][2]
            senti = sentiment(review)
            cmd = "update movie_reviews set sentiment=%s where movie_id=%s and user_name=%s and review = %s;"
            cursor.execute(cmd, (senti, movie_id, user_name, review))
            mydb.commit()
```

d.



```
In [11]: cmd = ''' select movie_title as movie_name
                   from movie_info natural join movie_reviews
                   where audience_rating>3.5 and sentiment="positive"
                   group by movie_id
                   having count(sentiment) >= 2
                   order by audience_rating desc
                   limit 5; '''
         cursor.execute(cmd)
         x=cursor.fetchall()
         for foo in x:
             print(foo)

         ('The Lord of the Rings: The Fellowship of the Ring',)
         ('Room',)
         ('The Man Who Shot Liberty Valance',)
         ('Duck Soup',)
         ('Run Lola Run',)
```

References :

[1] Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA,