**1 . Create Student Table with appropriate constraints. STUDENT(sno number primary key, sname text(20), age number, total\_marks number) write python programs to perform following task: 1) store the table data into a dataframe and display the dataframe. 2) List out top three records from the dataframe 3) Display all records from dataframe whose age is not less than 18. 4) Display age of student whose sno is 5. (use loc() and iloc() function)**

**< SQLite Code >**

sqlite> create table STUDENT(

(x1...> sno number primary key,

(x1...> sname text(20) not null,

(x1...> age number,

(x1...> total\_marks number);

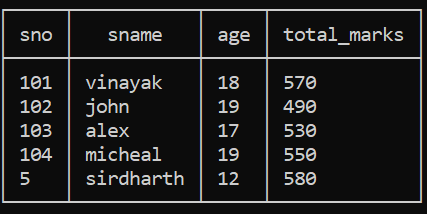
sqlite> insert into STUDENT values(101,'vinayak',18,570);

sqlite> insert into STUDENT values(102,'john',19,490);

sqlite> insert into STUDENT values(103,'alex',17,530);

sqlite> insert into STUDENT values(104,'micheal',19,550);

sqlite> select \* from Student;



**<PYTHON CODE>**

import sqlite3

import pandas as pd

try:

db = sqlite3.connect('jounralsolution')

db.cursor()

df = pd.read\_sql\_query('SELECT \* FROM STUDENT', db)

print("Printing entire dataframe........:")

print(df)

print("\nTask 2: Top three records:")

top\_three=df.head(3)

print(top\_three)

print("\nTask 3: Records with age not less than 18:")

above\_18=df[df['age']>=18]

print(above\_18)

print("\nTask 4: Displaying age of student whose sno is 5: ")

print("\nage of student with sno 5 (loc):",df.loc[df['sno']==5 ,'age'].values[0])

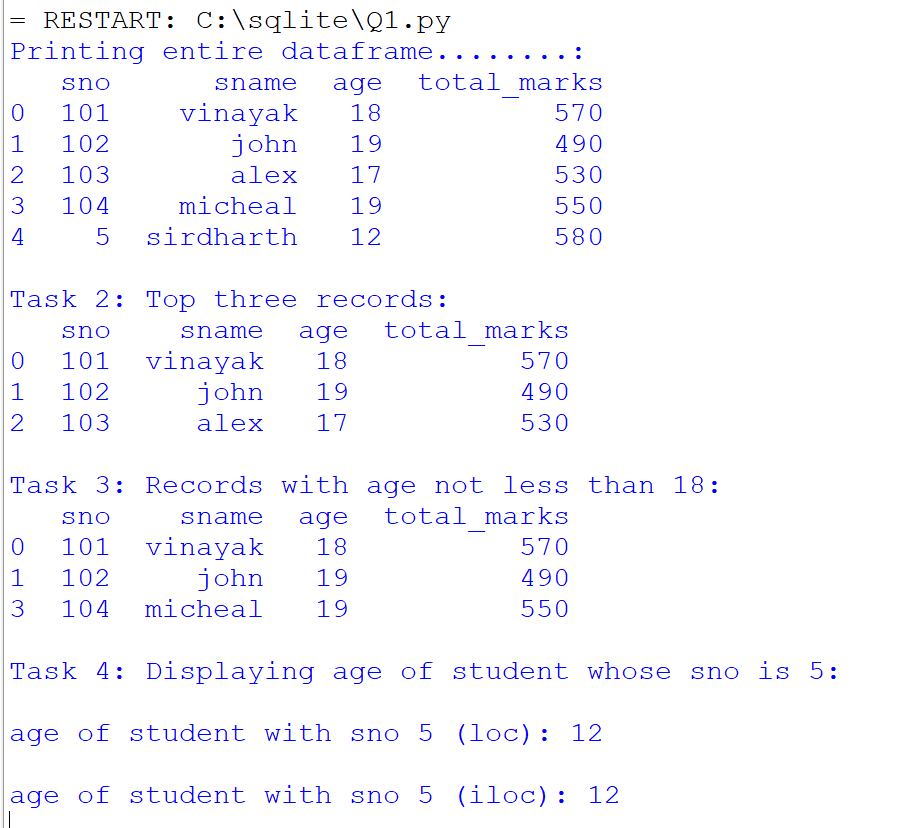
print("\nage of student with sno 5 (iloc):",df.iloc[df.index[df['sno']==5],df.columns.get\_loc('age')].values[0])

db.commit()

except:

print('ERROR 402: Something went wrong.....')

db.rollback()



**Q2. Create following table and store any five records: Employee(eno number primary key, Ename text(20),designation text(10),basic number , da number, gross\_salary number) write python programs to perform following tasks:**

**1) Store the table data into dataframe and display the dataframe. 2) Sort the dataframe based used on gross salary and List out bottom two record from the dataframe. 3) Display all records from dataframe whose gross Display gross salary is more than 25000 . 4) Display gross salary of employee whose eno is 4.**

**<SQLite Code>**

sqlite> CREATE TABLE IF NOT EXISTS Employee (

(x1...> eno INTEGER PRIMARY KEY,

(x1...> Ename TEXT(20),

(x1...> designation TEXT(10),

(x1...> basic REAL,

(x1...> da REAL,

(x1...> gross\_salary REAL

(x1...> );

sqlite> insert into Employee values (1, 'Vinayak', 'Manager', 50000, 10000, 60000);

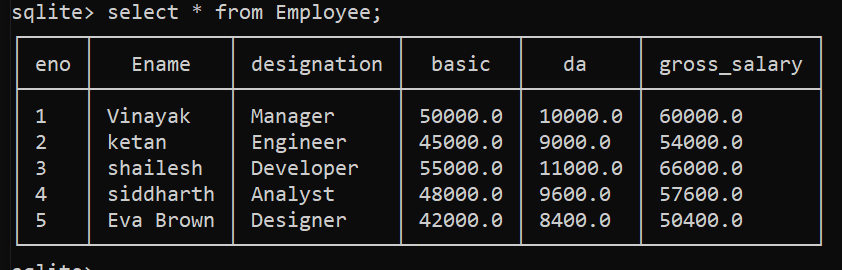
sqlite> insert into Employee values (2, 'ketan, 'Engineer', 45000, 9000, 54000);

sqlite> insert into Employee values (3, 'shailesh', 'Developer', 55000, 11000, 66000);

sqlite> insert into Employee values (4, 'siddharth', 'Analyst', 48000, 9600, 57600);

sqlite> insert into Employee values (5, 'Eva Brown', 'Designer', 42000, 8400, 50400);

sqlite> select \* from Employee;



**< PYTHON CODE >**

import sqlite3

import pandas as pd

try:

db = sqlite3.connect('jounralsolution')

db.cursor()

df = pd.read\_sql\_query('SELECT \* FROM Employee', db)

print("\nPrinting entire dataframe........:")

print(df)

print("\nTask 2:Sorting the dataframe based used on gross salary....")

df\_sorted = df.sort\_values(by='gross\_salary')

print(df\_sorted)

print("\nprinting bottom two record from the dataframe.....")

bottom = df.tail(2)

print(bottom)

print("\nTask 3:Displaying all records from dataframe whose gross Display gross salary is more than 25000")

sorted\_data = df[df['gross\_salary']>25000]

print(sorted\_data)

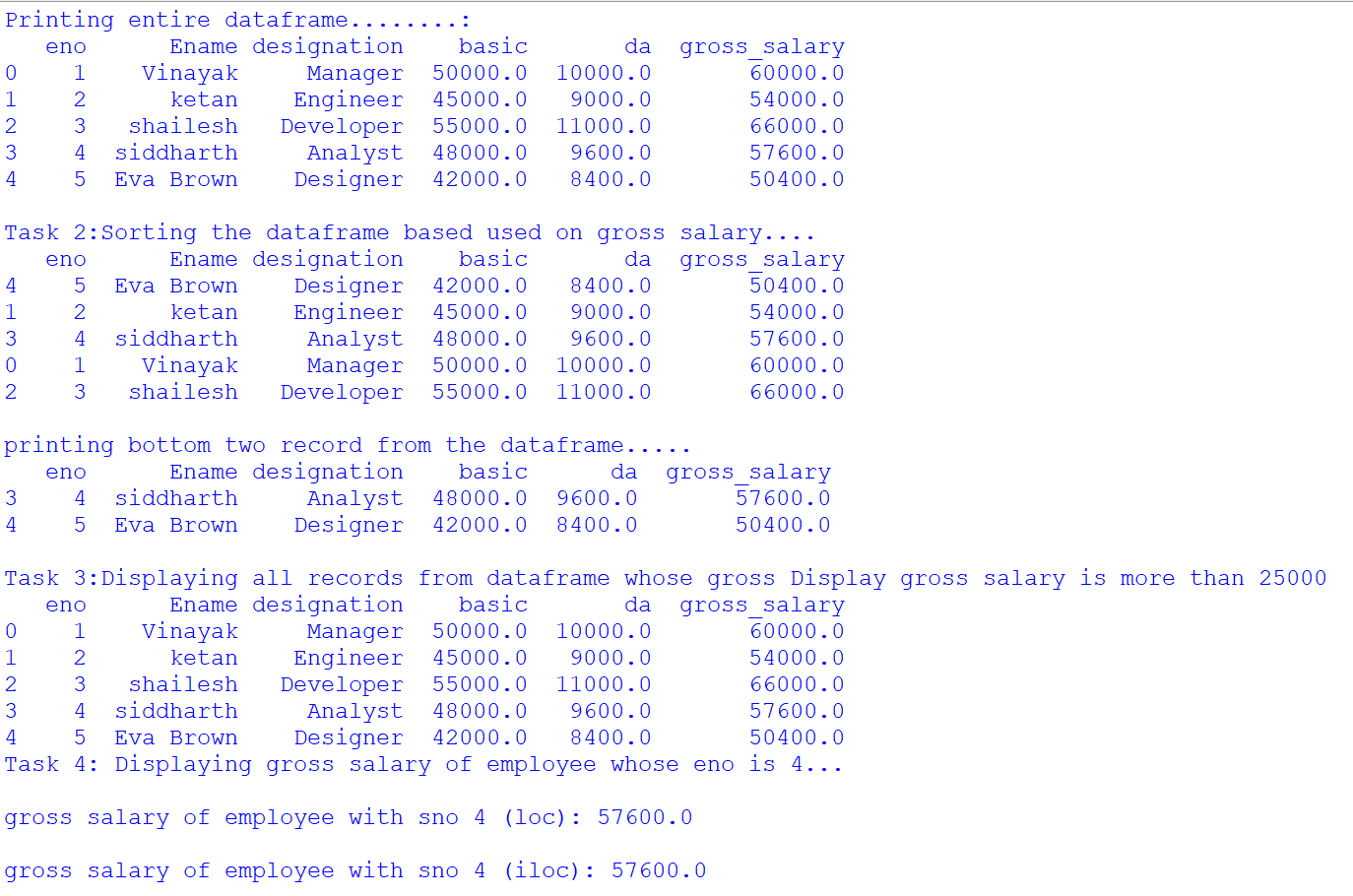
print("Task 4: Displaying gross salary of employee whose eno is 4...")

print("\ngross salary of employee with sno 4 (loc):",df.loc[df['eno']==4 ,'gross\_salary'].values[0])

print("\ngross salary of employee with sno 4 (iloc):",df.iloc[df.index[df['eno']==4],df.columns.get\_loc('gross\_salary')].values[0])

except:

print('Something went wrong')



**Q3. Create CSV file for product selling for 6 months and add only 5 record for 5 different product**

**Prod\_name JAN FEB MAR APR MAY JUN**

**Create Python script for following program:**

**1.Read data into DataFrame 2) Add columns and calculate total\_sell, average\_sell 3) Plot Total sell and average sell together on Line chart with proper legends, Titles and Lables. 4) Save the DataFrame to CSV named 'sell\_analysis.csv'**

**< SQLiteCode >**

sqlite> CREATE TABLE IF NOT EXISTS sales (

(x1...> sid INTEGER PRIMARY KEY AUTOINCREMENT,

(x1...> year INTEGER NOT NULL,

(x1...> totalsales REAL,

(x1...> CONSTRAINT year\_check CHECK (year >= 0),

(x1...> CONSTRAINT totalsales\_check CHECK (totalsales >= 0)

(x1...> );

sqlite> insert into sales values(1,2018,53000.33);

sqlite> insert into sales values(2,2019,24000.56);

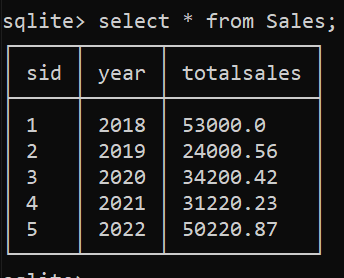
sqlite> insert into sales values(3,2020,34200.42);

sqlite> insert into sales values(4,2021,31220.23);

sqlite> insert into sales values(5,2022,50220.87);

sqlite> .mode box

sqlite> select \* from sales;



**< PYTHON CODE >**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('data.csv')

print(df)

print("\nTask 2: AddING columns and calculate total\_sell and average\_sell...")

df['total\_sell']=df.iloc[:,1:].sum(axis=1)

df['average\_sell']=df.iloc[:,1:].mean(axis=1)

print(df)

print("\nTask 3: printting linediagram......")

plt.figure(figsize=(10,6))

plt.plot(df['prod\_name'],df['total\_sell'],label='Total Sell',marker='o')

plt.plot(df['prod\_name'],df['average\_sell'],label='Average Sell',marker='o')

plt.xlabel('products')

plt.ylabel('amount')

plt.title('product sell analysis')

plt.legend()

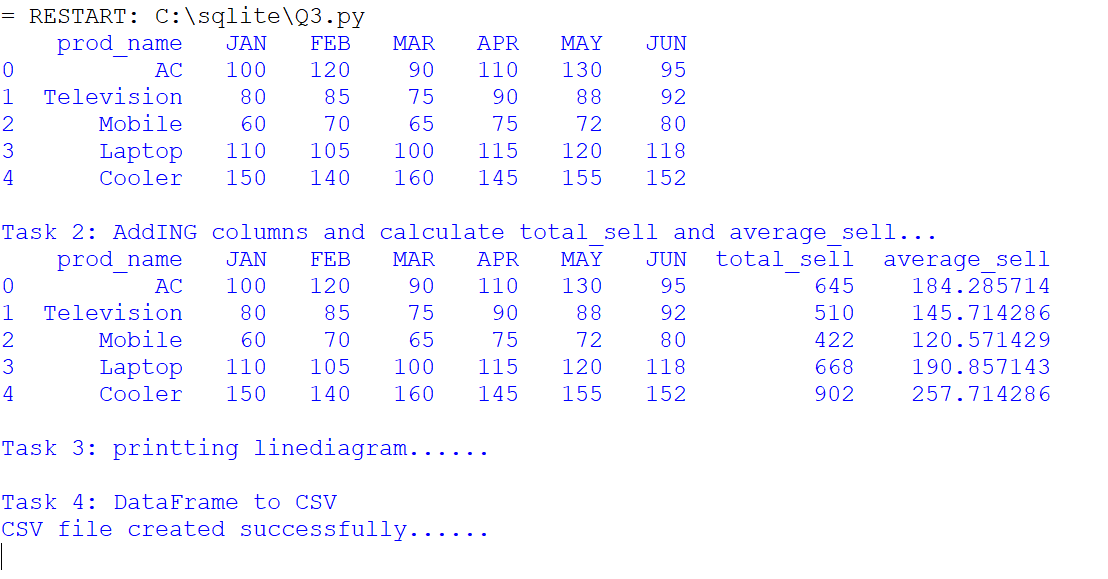
plt.grid(True)

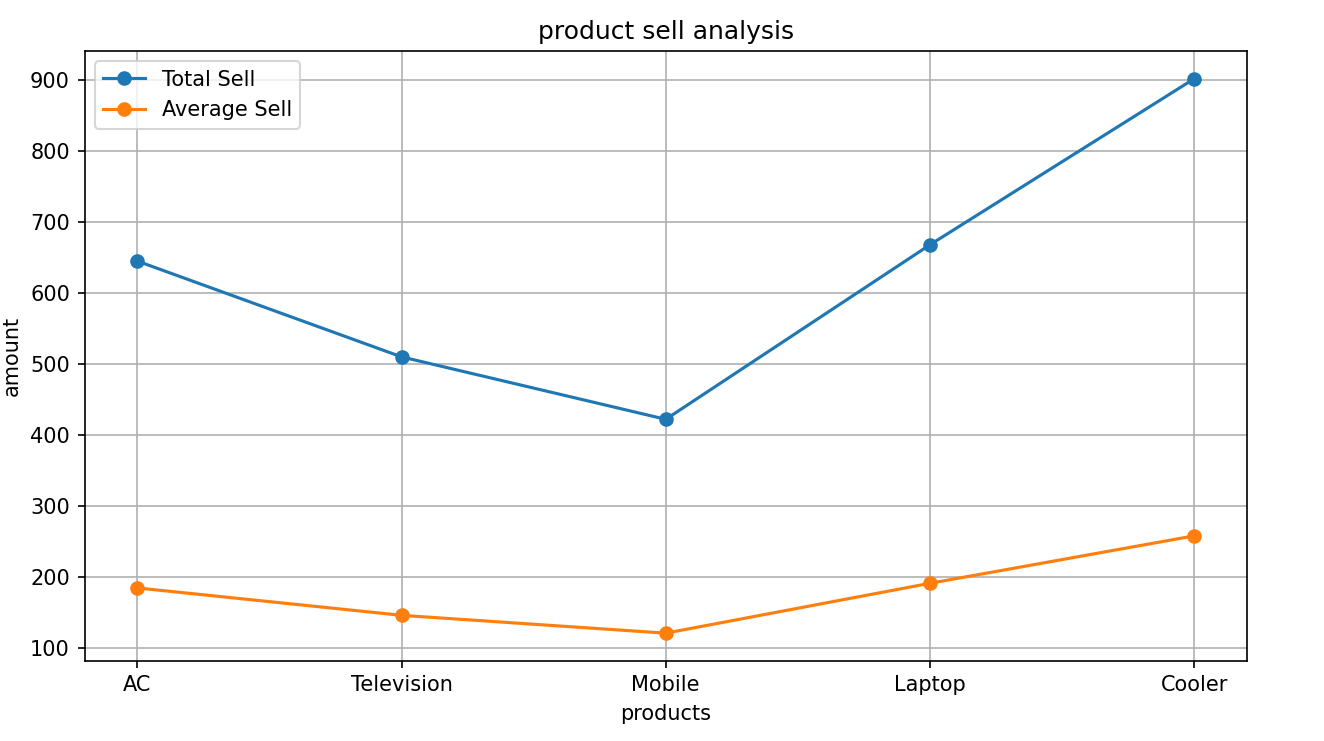
plt.show()

print("\nTask 4: DataFrame to CSV")

df.to\_csv('sell\_analysis.csv',index=False)

print("CSV file created successfully......")





**Q4. Write a phython script to do following on student (Rollno, Name, Sub 1, Sub 2, Sub 3, total) table:**

**1) Insert atleast 5 to 10 records. 2) Update the specific record value. 3) Delete the record specific record. 4) Display student detail who got highest total marks**

**< PYTHON CODE >**

import sqlite3

db = sqlite3.connect('jounralsolution')

try:

cur=db.cursor()

cur.execute("drop table if exists student1;")

cur.execute("""CREATE TABLE IF NOT EXISTS Student1 (

Rollno INTEGER PRIMARY KEY,

Name TEXT,

Sub1 INTEGER,

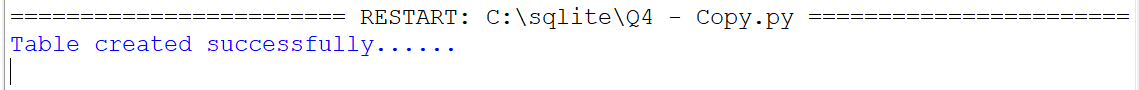
Sub2 INTEGER,

Sub3 INTEGER,

Total INTEGER

);""")

print('Table created successfully......')



data = [(1, ‘Vinayak’, 85, 90, 78, 253),

(2, 'Virat', 92, 88, 76, 256),

(3, 'Mahendra', 78, 85, 80, 243),

(4, 'Rohit', 90, 92, 85, 267),

(5, 'Aakash', 88, 86, 94, 0)]

cur.executemany("""

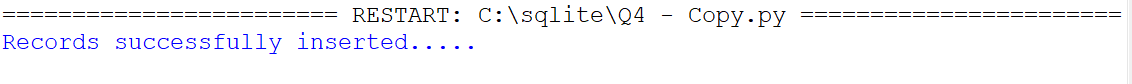
INSERT INTO Student1 (Rollno, Name, Sub1, Sub2, Sub3, Total)

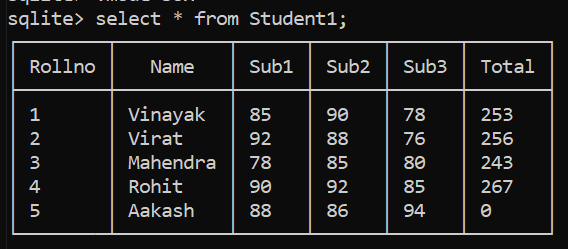
VALUES (?, ?, ?, ?, ?, ?)

""", data)

db.commit()

print("Records successfully inserted.....")



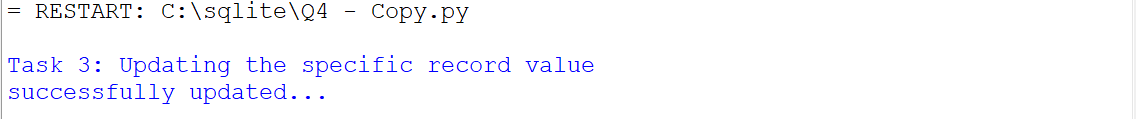


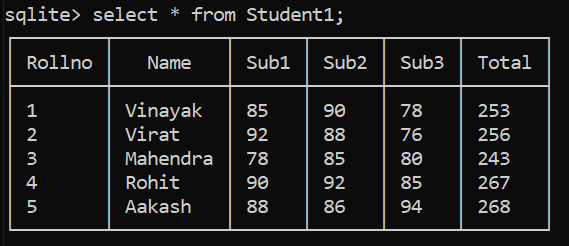
print("\nTask 3: Updating the specific record value")

cur.execute("UPDATE Student1 SET Total = ? WHERE Rollno = ?", ((88 + 86 + 94), 5))

print("successfully updated...")

db.commit()





print("\nTask 4: Delete the record specific record")

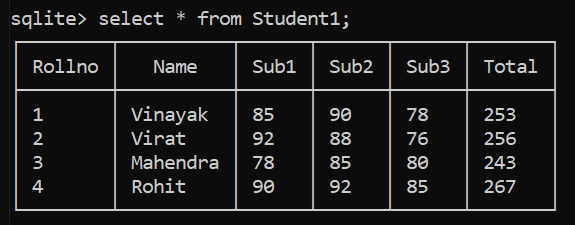
rno=int(input("Enter the rollno to be deleted: "))

cur.execute("delete from student1 where Rollno = ?",(rno,))

print("record deleted successfully....")

db.commit()





print("\nTask 5: Printing higest numbet std Details......")

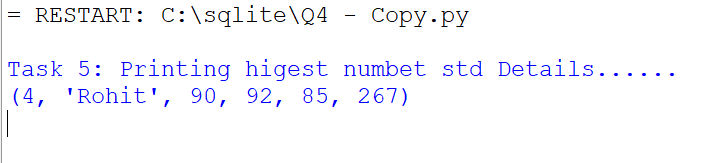
cur.execute("SELECT \* FROM Student1 WHERE Total = (SELECT MAX(Total) FROM Student1);")

maxnum=cur.fetchone()

print(maxnum)

except:

print("Something went wrong......")



**Q5. Write Python Script to do followings on item.csv (Item\_no, Item\_name, Price, Qty, total)**

**1) Write item's detail in the item.csv file. Calculate total = price \* Qty 2) Using data frame display item name and price whose price is between 1000 to 5000. 3) Display alternate records from item.csv file. 4) Display items whose price is minimum, maximum. 5) Sort the data according to item name wise. 6) Display items rows between 3th to 7th row. 7) Display last 6 rows**

**< PYTHON CODE >**

import pandas as pd

print("\nTask 1: Write item's detail in the item.csv file and calculate total")

data = {

'Item\_no': [1, 2, 3, 4, 5],

'Item\_name': ['Laptop', 'Blender', 'Scooter', 'TV', 'Freedge'],

'Price': [1500, 3000, 2500, 4500, 2000],

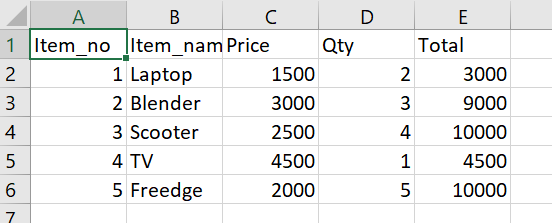
'Qty': [2, 3, 4, 1, 5]

}

df = pd.DataFrame(data)

df['Total'] = df['Price'] \* df['Qty']

df.to\_csv('item.csv', index=False)

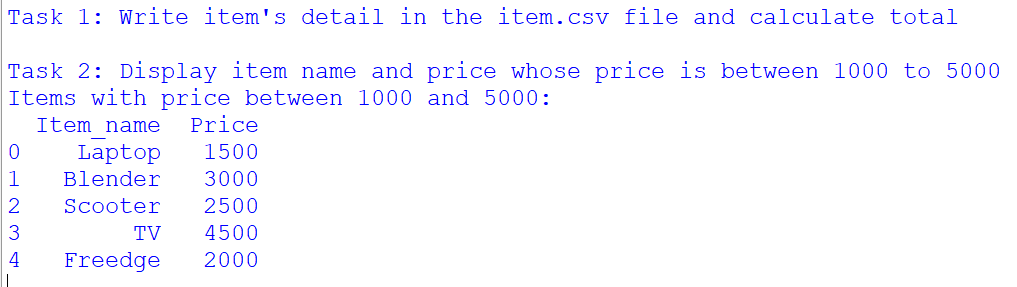


print("\nTask 2: Display item name and price whose price is between 1000 to 5000")

filtered\_df = df[(df['Price'] >= 1000) & (df['Price'] <= 5000)]

print("Items with price between 1000 and 5000:")

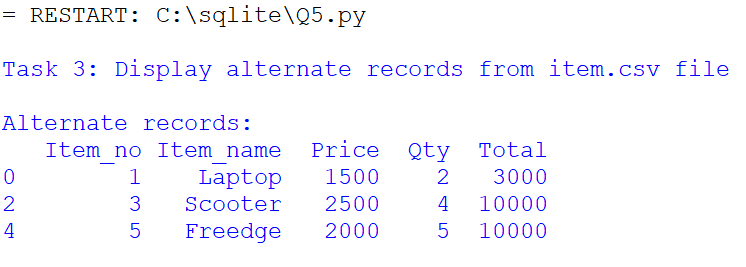
print(filtered\_df[['Item\_name', 'Price']])



print("\nTask 3: Display alternate records from item.csv file")

print("\nAlternate records:")

print(df[::2])



print("\nTask 4: Display items whose price is minimum and maximum")

min\_price\_item = df[df['Price'] == df['Price'].min()]

max\_price\_item = df[df['Price'] == df['Price'].max()]

print("\nItems with minimum price:")

print(min\_price\_item)

print("\nItems with maximum price:")

print(max\_price\_item)

print("\nTask 5: Sort the data according to item name wise...")

sorted\_df = df.sort\_values(by='Item\_name')

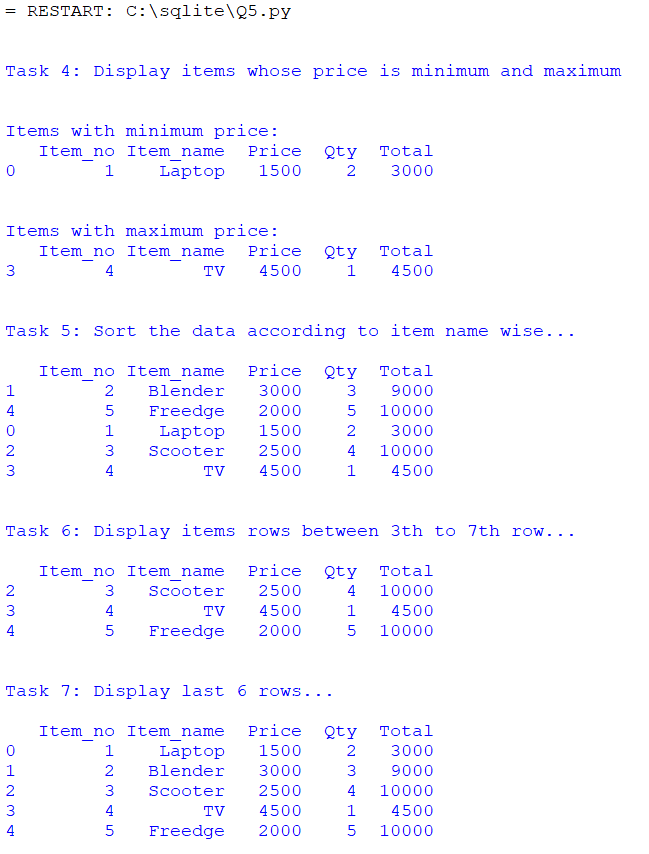
print(sorted\_df)

print("\nTask 6: Display items rows between 3th to 7th row...")

print(df.iloc[2:7])

print("\nTask 7: Display last 6 rows...")

print(df.tail(6))



**Q6. Sales (sid, year, totalsales) Create above table into a SQLite database with appropriate constraints. 1) Insert at least 5-10 records into the sales table 2) Export sales table data into sales.csv file. 3) Write a python scripts that read the sales.csv file and plot a bar chart that shows totalsales of the year. Also decorate the chart with appropriate title, lables, colours etc.**

**< SQLite Code >**

sqlite> CREATE TABLE sales (

(x1...> sid INTEGER PRIMARY KEY AUTOINCREMENT,

(x1...> year INTEGER NOT NULL,

(x1...> totalsales REAL);

sqlite> INSERT INTO Employee VALUES(1,'Vinayak','Manager',50000.0,10000.0,60000.0);

sqlite> INSERT INTO Employee VALUES(2,'Jane Smith','Engineer',45000.0,9000.0,54000.0);

sqlite> INSERT INTO Employee VALUES(3,'shailesh','Developer',55000.0,11000.0,66000.0);

sqlite> INSERT INTO Employee VALUES(4,'siddharth','Analyst',48000.0,9600.0,57600.0);

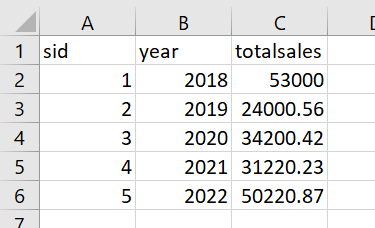
sqlite> INSERT INTO Employee VALUES(5,'Eva Brown','Designer',42000.0,8400.0,50400.0);

sqlite> .header on

sqlite> .mode csv

sqlite> .output sales.csv

sqlite> select \* from sales;



**< PYTHON CODE >**

import pandas as pd

import matplotlib.pyplot as plt

df=pd.read\_csv('sales.csv')

print(df)

plt.figure(figsize=(10,6))

plt.bar(df['year'],df['totalsales'],color='skyblue')

plt.xlabel('year')

plt.ylabel('total sales')

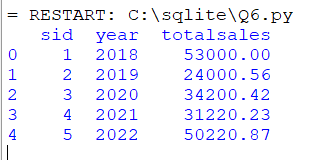
plt.title('totalsales by year')

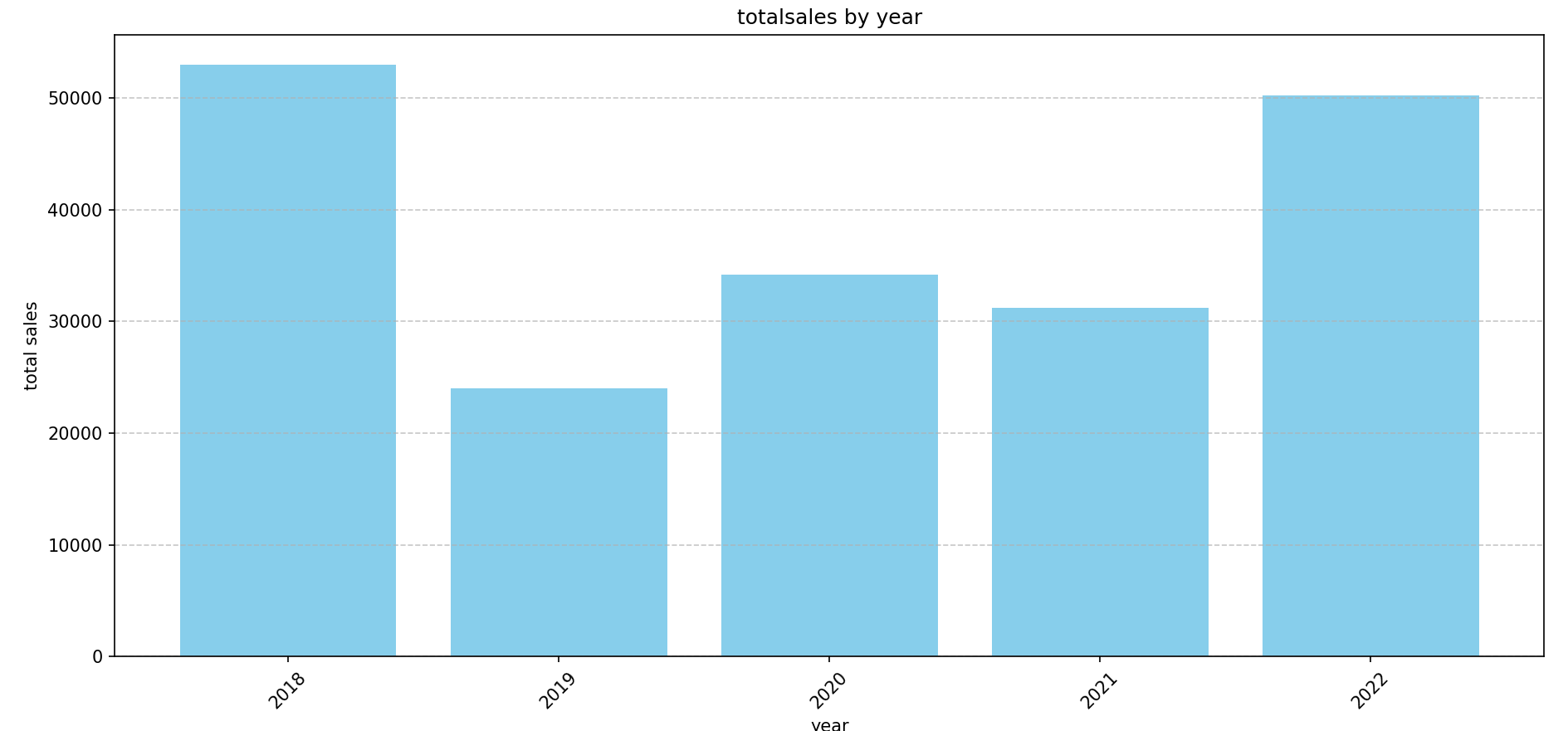
plt.xticks(rotation=45)

plt.grid(axis='y',linestyle='--',alpha=0.7)

plt.tight\_layout()

plt.show()





**Q7. Create following table with appropriate constraints in Collage Database: Employee (E\_ID, Name, Dob, Designation, Salary ) a) Dump Employee table structure and data in Emp.csv file. b) Dump whole Database named College in Emp1.csv file.**

sqlite> CREATE TABLE Employee (

(x1...> E\_ID INTEGER PRIMARY KEY,

(x1...> Name TEXT NOT NULL,

(x1...> Dob DATE,

(x1...> Designation TEXT NOT NULL,

(x1...> Salary REAL CHECK (Salary >= 0)

(x1...> );

sqlite> INSERT INTO Employee VALUES(1,'Amrit','1990-01-15','Manager',60000.0);

sqlite> INSERT INTO Employee VALUES(2,'Suresh','1995-03-22','Engineer',55000.0);

sqlite> INSERT INTO Employee VALUES(3,'kartik','1987-07-10','Analyst',50000.0);

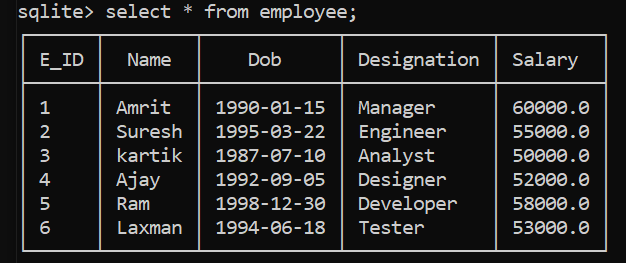
sqlite> INSERT INTO Employee VALUES(4,'Ajay','1992-09-05','Designer',52000.0);

sqlite> INSERT INTO Employee VALUES(5,'Ram','1998-12-30','Developer',58000.0);

sqlite> INSERT INTO Employee VALUES(6,'Laxman','1994-06-18','Tester',53000.0);

sqlite> .mode box

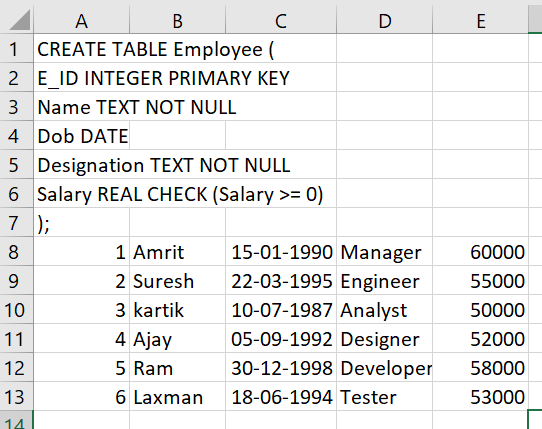
sqlite> select \* from Employee;



sqlite> .mode csv

sqlite> .output Emp.csv

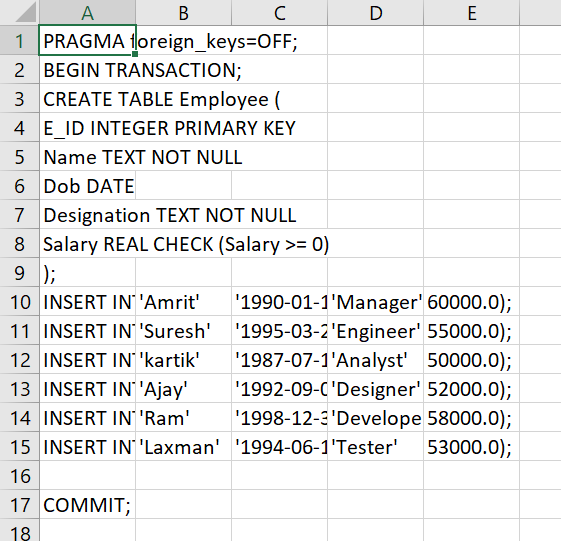
sqlite> .schema Employee



sqlite> select \* from Employee;

sqlite> .output Emp1.csv

sqlite> .dump



**8. Create following table with appropriate Constraints: Product (prod\_id , prod\_name , price, qty,total\_amount) 1) Import Product.csv file data into Product table. 2) Export Product table data into prod.csv file.**

**< SQLite Code >**

sqlite> CREATE TABLE Product (

(x1...> prod\_id TEXT PRIMARY KEY,

(x1...> prod\_name TEXT NOT NULL,

(x1...> price real NOT NULL,

(x1...> qty INTEGER not null,

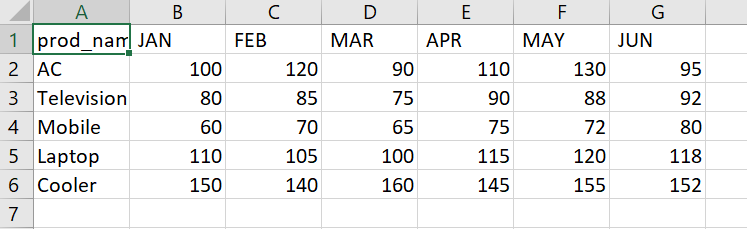
(x1...> total\_amount REAL);

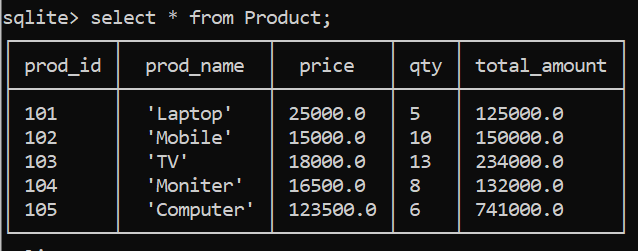
sqlite> .mode csv

sqlite> .import product.csv Product

sqlite> .mode box

sqlite> select \* from Product;

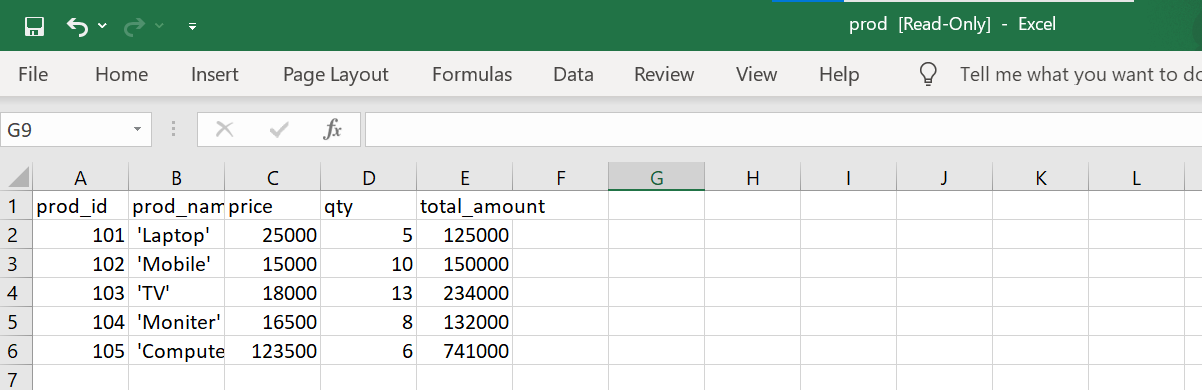
****

****

sqlite> .mode csv

sqlite> .output prod.csv

sqlite> select \* from Product;



**Q9. Employee(Eno number ,Ename text ,Desg text ,Salary number ,City text ,Email text) Write a SQL trigger named emp\_trigger that is designed to execute before inserting records into the emp table. The trigger should perform the following action: 1) Check if the 'email' field in the newly inserted record follows a specific email address pattern. (example :** [**abc@gmail.com**](mailto:abc@gmail.com)**)**

**< SQLite Code >**

sqlite> CREATE TABLE Employees (

(x1...> Eno INTEGER PRIMARY KEY,

(x1...> Ename TEXT NOT NULL,

(x1...> Desg TEXT,

(x1...> Salary REAL,

(x1...> City TEXT,

(x1...> Email TEXT

(x1...> );

sqlite> create trigger check\_email

...> before

...> insert on Employees

...> begin

...> select

...> case

...> when new.email not like '%\_@\_%.\_%'

...> then

...> raise(abort,'Invail Email Address')

...> end;

...> end;

sqlite> .mode box

sqlite> select \* from Employees;

