1.Create ‘Sales’ database.

.open sales.db

2. Create tables and insert records in ‘Sales’ database and perform SQLite commands given in Practical Assignment 1

Salesman (sid, name, city, commission)

Order (oid, pur_amt, ord_date, sid)

Customer(cid,cust_name,grade,sid)

---create tables---

create table salesmen(sid primary key,name,city,commission);

---insert data---

insert into salesmen values(5001,'mohan patel','anand',0.15);
insert into salesmen values(5002,'nail shah','surat',0.13); insert into salesmen values(5005,'preet vyas','ahmendabad',0.11); insert into salesmen values(5006,'jeevanmehta','navsari',0.14); insert into salesmen values(5003,'paul adam','nadiad',0.12); insert into salesmen values(5007,'ramesh patel','surat',
0.13);

---create tables---

create table order(oid primary key,pur_amt,ord_date,sid references salesmen(sid));

---insert data---

insert into order values(1,12000,'5-des-2020',5001);

insert into order values(2,42000,'20-des-2020',5002);

insert into order values(3,2000,'2-feb-2021',5005); insert into order values(4,14000,'23-mar-2021',5005); insert into order values(5,23000,'15-apr-2021',5003); insert into order values(6,33000,'20-may-2021',5001); insert into order values(7,32000,'15-jun-2021',5003); insert into order values(8,23500,'22-jun-2021',5003); insert into order values(9,43000,'1-july-2021',5001); insert into order values(10,12000,'5-july-2021',5002);

---create tables---

create table customer(cid primary key,name,grade,sid references salesmen(sid));

---insert data---

insert into customer values('c1','mohit patel','100',5001);

insert into customer values('c2','geeta vyas',200,5003);

insert into customer values('c3','jaya patil',100,5005); insert into customer values('c4','visal gohel',300,5005); insert into customer values('c5','kartik goyenka',200,5002); insert into customer values('c6','meera prajapati',100,5001); insert into

customer values('c7','veer vyas',300,5002); insert into

customer values('c8','maya mehta',200,5003);

5 performing following tasks:

5.1 display customer's details whose grade is 200.

select * from customer where grade=200;

5.2 display salesman's details whose name start with 'p'.

select * from salesmen where name like 'p%';

5.3 display unique grade of customers.

select distinct grade from customer where grade in(300);

5.4 display order's details whose purchase amount is in between 10000 to 20000.

select * from order where pur_amt>10000 and pur_amt<20000;

5.5 display customer's details of sid 5001 and 5003.

select * from customer where sid in(5001,5003);

5.6 display the top three order details.

select * from order limit 3;

5.7 display the top 5th,6th and 7th order's details.

select * from order order by oid limit 3 offset 4;

5.8 display salesman's details who doesn't have any customer.

select * from salesmen where sid not in(select sid from customer);

5.9 display salesman ID wise total purchase amount.

select s.sid,sum(pur_amt)from salesmen s,order o where s.sid=o.sid group by s.sid;

5.10 display salesman ID wise details whose average grade is more then 150.

select s.sid,avg(grade) from salesmen s,customer c where s.sid=c.sid group by s.sid having avg(grade)>150;

5.11 display the order detail in descending order as per purchase amount.

select * from order order by pur_amt desc;

5.12 display customer's detail with bonus.calculate bonus as per follwing criteria.

grade=100 -> bonus=5000

grade=200 -> bonus=10000

otherwise -> bonus=15000

select cid,name,grade,sid,

case

```
        when grade=100 then 5000
when grade=200 then 10000
else 15000    end as bonus
        from
        customer;
```

5.13 display order detail with class. class is declared as per follwing criteria.

SID=5001 or 5002 -> class='A'

SID=5003 or 5005 -> class='B' otherwise

-> class='C'

```
        select oid,pur_amt,ord_date,sid,
        case
        when sid=5001 or sid=5002 then
'A'     when sid=5003 or sid=5005 then
'B'     else 'C'        end as class
        from
order;
```

5.14 display customer's detail with salesman name.

```
        select c.*,s.name from customer c,salesmen s where s.sid=c.sid;
```

5.15 display order details with salesman details.

```
select o.*,s.* from order o,salesmen s where o.sid,s.sid;
```

5.16 display order details with salesman commision.

```
select o.*,s.commission from order o,salesmen s where o.sid=s.sid;
```

3. Create database trigger as follow Before inserting new record in Salesman table check whether commission is in between 0 or 1 or not.

If not display error message.

```
create trigger trg_commission
before insert on salesmen
begin
select
case
when new.commission not between 0 and 1
then raise(abort,'invalid commission')
end;
end;
```

4. Check whether inserted or update order_date in Order table is in 'yyyy-mm-dd' format or not. If not generate error message.

```
        --before insert


        create trigger trg_ordete
before insert on order
        begin
select
case
        when new.ord_date not like '____-__-__'
then raise(abort,' not invalid date formate') end;
end;

        --before update


        create trigger trg_ordete1
before update on order
        begin
select
case
 when new.ord_date not like '____-__-__'  then
raise(abort,' not invalid date formate')
        end;
end;
```

5. Create one TRANSACTION_LOG table (ID, table_name, current_date) when new record is inserted into Salesman or Customer tables. (2 triggers).

--(1)--

```
create table tbl_log(sid,tbl_name,current_date);

drop trigger trg_trans_log;
create trigger trg_trans_log
before insert on salesmen
begin
  insert into tabl_log values(new.sid,'salesmen',datetime('now')); end;
```

--(2)--

```
create table tbl_log(cid,tbl_name,current_date);

drop trigger trg_trans_log;
create trigger trg_trans_log
before insert on customer
begin
        insert into tbl_log values(new.cid,'customer',datetime('now'));
```

```
        end;
```

6. Example: TRANSACTION_LOG.

```
        create table trans(sid,name,div,class);

        drop trigger trg_trans_log1;
create trigger trg_trans_log1
        after insert
        on tblstud
begin
                insert into trans values(new.sid,new.name,new.div,new.class);
        end;
```

7. Create one AUDIT table (cid, old_name, new_name, old_sid, new_sid, operation, date) for Customer table's modified operations (For Update and delete). (2 triggers)

```
        --(1)--
        create table audit(sid,old_name,new_name,old_sid,new_sid,operation,date);

        drop trigger trg_audit;
```

```
create trigger trg_audit
before update on customer
begin
        insert into audit values(old.sid,old.name,new.name,old.sid,new.sid,
'update',datetime('now'));
    end;
```

--(2)--

```
create table audit(cid,old_name,new_name,old_sid,new_sid,operation,date);

drop trigger trg_audit;
create trigger trg_audit
before delete on
customer begin
        insert into audit
values(old.cid,old.name,old.name,old.sid,old.sid,datetime('now'),'delete');
    end;
```

13. If any record is deleted from Salesman table then first check whether related records are available in child table (customer). If available then generate error message and don't allow to delete record.

```
        drop  trigger  trg_delete;
        create trigger trg_delete
        before delete
        on
salesmen      begin
select        case
        when old.sid in(select old.sid from customer ) then
RAISE(ABORT,"CHILD record contain parent table record")
        end;
end;
```

14. Perform operation for dumping data into files.

7.1--dump entire 'SALES' database into file name 'sales.sql'.

```
        .output sales.sql
        .dump
```

7.2--dump customer tables's information into 'customer.txt' file.

```
        .output customer.txt
        .dump customer
```

7.3--append order table's information into same file 'customer.txt'.

.open customer.txt

.dump order

7.4--dump all table structures into 'sales_schema.sql' file.

.output sales_schema.sql

.schema

7.5--dump data of all tables into 'sales_data.txt' file.

.output sales_data.txt select *

from salesmen; select * from

customer; select * from order;

15. Create “Calculator.py” module and write mathematical functions.

def addnum(n1,n2):

return(n1+n2) def

subnum(n1,n2):

return(n1-n2) def

mulnum(n1,n2):

return(n1*n2)

16. Create “calculation.py” file. Take input of required numbers and call each function of “Calculator.py” module and display appropriate message.

```
import calculator
n1=int(input("enter first no:"))
n2=int(input("enter second no:"))
```

```
        a=calculator.addnum(n1,n2)
print(a)        a=calculator.subnum(n1,n2)
print(a)        a=calculator.mulnum(n1,n2)
print(a)


        --message--
        enter first no:10
enter second no:5
        15
        5
        50
```

17.

1.1. Create “Sales.db” database and perform queries given in Assignment 2 Question 3.4 to 3.17

```
        --create database--
        .open sales.db
```

3.4 diaplay 1st and 2nd record using fetchone() method.

```
        import sqlite3
conn=sqlite3.connect('sales.db')
a=conn.execute('select * from salesmen')
print(a.fetchone())    print(a.fetchone())
```

3.5 display data using fetchall() method.

```
import sqlite3
conn=sqlite3.connect('sales.db')
a=conn.execute('select * from salesmen')
print(a.fetchall())
```

3.6 display 4th and 5th records using fetchall() method.

```
import sqlite3
conn=sqlite3.connect('salesmen.db')
print(conn) row=conn.execute(f'select *
from salesmen') a=row.fetchall() print('-'*60)
print(a[3][0],a[3][1],a[3][2],a[3][3])
print(a[4][0],a[4][1],a[4][2],a[4][3])
```

3.7 display data without using fetchone() and fetchall() method.

```
import sqlite3
conn=sqlite3.connect('sales.db')
row=conn.execute('select * from salesmen')      for i
in row:
        print(i)
```

3.8 display record of salesman who are leaving in surat city. import sqlite3

```
conn=sqlite3.connect('sales.db') row=conn.execute(“select *
from salesmen where city=’surat’”) for i in row:
    print(i)
```

3.9 display record of salesman whose name consists letter "M".

```
import sqlite3
conn=sqlite3.connect('sales.db')   row=conn.execute('select * from
salesmen where name like "%m%"')       for i in row:
    print(i)
```

3.10 display record of salesman whose commission is in range of 0.11 to 0.13.

```
import sqlite3
conn=sqlite3.connect('sales.db')
row=conn.execute(“select * from salesmen where commission between
0.11 and 0.13”)       for i in row:
    print(i)
```

3.11 display record of salesman of anand and navsari city.

```
import sqlite3
```

```
conn=sqlite3.connect('sales.db')  row=conn.execute('select * from
salesmen where city in('anand','navsari'))        for i in row:
        print(i)
```

3.12 take salesman's SID from the user and display his details.

```
import sqlite3
conn=sqlite3.connect('sales.db')  sid=int(input('enter SID:'))
row=conn.execute(f"select * from salesmen where sid={sid}")
print(row.fetchall())
```

3.13 modify the commission of 5006 to 0.15.

```
import sqlite3
conn=sqlite3.connect('sales.db')
conn.execute(f"update salesman set commission=0.15 where
sid=5006")  print('record successfully updated')      conn.commit()
```

3.14 modify city to bharuch whose commission is 0.13.

```
import sqlite3
conn=sqlite3.connect('sales.db')
```

conn.execute(f"update salesman set commission=0.13 where city='bharuch'") print('record successfully updated') conn.commit()

3.15 delete the record of 5007.

```
import sqlite3
conn=sqlite3.connect('sales.db') conn.execute(f"delete
from salesmen where sid=5007") print('record
successfully deleted') conn.commit()
```

3.16 delete the record of salesman of bharuch city.

```
import sqlite3
conn=sqlite3.connect('sales.db')
conn.execute(f"delete from salesmen where city='bharuch'") print('record successfully deleted')
conn.commit()
```

3.17 insert three record into salesmen table as per user's input.

```
import sqlite3
conn=sqlite3.connect('sales.db')
print(conn) for i in range(3):
```

```
                    sid=int(input('enter the sid--'))
name=input('enter name--')                    city=input('enter
city--')                    commission=float(input('enter
commission--'))
        .execute(f"insert into salesmen values{sid,name,city,commission}")
        conn.commit()
        row=conn.execute(f'select * from salesmen') for
        i in row:
            print(i)
```

1.2. Create following table and insert records.

```
        --create table--

        create table tblsales(sid primary key,name,city,commision);
```

1.3. Salesman (sid, name, city, commission) (primary key = SID)

```
        --insert records--

        insert into tblsales  values(5001,"Mohan patel","Anand",0.15);
insert into tblsales  values(5002,"Nail Shah","Surat",0.13);  insert into
tblsales  values(5005,"Preet Vyas","Ahemdabad",0.11);  insert into
tblsales  values(5006,"Jeevan Mehta","Navasari",0.14);  insert into
```

tblsales values(5003,"Paul Adam","Nadiad",0.12); insert into tblsales values(5007,"Ramesh patel","Surat",0.13);

18. Write examples of user defined package and import package in python file.

--user define package--

package name==>practical

module 1==>p1

```
def add(n1,n2):        return
n1+n2 module 2==>p2 def
mul(n1,n2):
                return n1*n2
```

--import package--

```
import practical.p1,practical.p2
n1=int(input('enter n1--'))
n2=int(input('enter n2--'))
a=practical.p1.add(n1,n2)
b=practical.p2.mul(n1,n2)   print(a)
print(b)
```

19. Create ‘stud.csv’ file which includes ‘rno, name, sub1, sub2, sub3, sub4, sub5’ fields. Retrieve data from above created tables.

```
import csv     open
with("stud.csv","r") as f1:
        r1=csv.reader(f1)
        print('-'*60)
        for row in r1:
                print(row)
```

20.

2. Extract data from ‘stud.csv’ file and store it as dataframe. Perform following tasks:

```
--file store as dataframe--
import pandas as pd
df=pd.read_csv(‘stud.csv’)
print(df)
```

2.1. Display line chart which includes rollno on x axis and Sub1’s marks on y axis.

2.1.1. Give proper headings.

```
import pandas as pd
```

```
import matplotlib.pyplot as
plt df=pd.read_csv('p1.csv')
print(df) a=df.rno b=df.sub1
plt.plot(a,b,'m--')
plt.xlabel('roll no')
plt.ylabel('sub1')
plt.title("student's result")
plt.show()
```

2.1.2. Display grid.

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("stud.csv")
print(df) a=df.rollno
```

```
        b=df.sub1
        plt.plot(a,b,'m--',linewidth=2)
plt.xlabel('roll no')
plt.ylabel('sub1')
plt.title(“student's marks”)
plt.grid()      plt.show()
```

2.2. Display comparison line chart for Sub1, Sub2, Sub3, Sub4 and Sub5 with respect to Rollno.

2.2.1. Give proper heading.

```
import pandas as pd    import
matplotlib.pyplot      as      plt
df=pd.read_csv("stud.csv")  print(df)
a=df.rollno  s1=df.sub1  s2=df.sub2
s3=df.sub3 ` s4=df.sub4  s5=df.sub5
        plt.plot(a,s1,'m--',linewidth=2)
plt.plot(a,s2,'m--',linewidth=2)      plt.plot(a,s3,'m-
-',linewidth=2)        plt.plot(a,s4,'m--',linewidth=2)
plt.plot(a,s5,'m--',linewidth=2)      plt.xlabel('roll
no')
```

303-DB-PYTHON PRACTICAL ASSIGNMENT

```
plt.ylabel('marks')

plt.title('student's marks')

plt.show()
```

2.2.2. Display legend.

```
import pandas as pd import
matplotlib.pyplot as plt
df=pd.read_csv("stud.csv")
print(df) a=df.rollno
s1=df.sub1 s2=df.sub2
s3=df.sub3
```

```
`       s4=df.sub4 s5=df.sub5
        plt.plot(a,s1,'m--',linewidth=2)
        plt.plot(a,s2,'m-',linewidth=2)
        plt.plot(a,s3,'m-.',linewidth=2)
        plt.plot(a,s4,'m:',linewidth=2)
        plt.plot(a,s5,'m--',linewidth=2)
        plt.xlabel('roll no')
        plt.ylabel('marks')
        plt.title('student's marks')
        plt.legend(['sub1','sub2','sub3',' 
        plt.show()
                                              sub4','sub5'])
```

2.2.3. Display each line with diff formatting.

303-DB-PYTHON PRACTICAL ASSIGNMENT

```
import pandas as pd import matplotlib.pyplot as plt
df=pd.read_csv("stud.csv") print(df)
a=df.rollno s1=df.sub1 s2=df.sub2
s3=df.sub3 ` s4=df.sub4 s5=df.sub5
        plt.plot(a,s1,'m--',linewidth=2)
plt.plot(a,s2,'b-',linewidth=2)
plt.plot(a,s3,'r-.',linewidth=2)
plt.plot(a,s4,'k:',linewidth=2)
plt.plot(a,s5,'y--',linewidth=2)
```

plt.xlabel('roll no') plt.ylabel('marks')

plt.title('student's marks') plt.show()

2.3. Display above two charts in one figure (Use subplot columnwise)

```
import pandas as pd
import matplotlib.pyplot as
plt
df=pd.read_csv('stud.csv')
print(df) a=df.rno
```

```
s1=df.sub1 s2=df.sub2
plt.subplot(1,2,1)
plt(a,s1,'m--')
plt.xlabel('roll no')
plt.ylabel('marks')
plt.title("student's
result") plt.subplot(1,2,2)
plt.plot(a,s2,'y-')
plt.xlabel('roll no')
plt.ylabel('marks')
plt.title("student's
result") plt.grid()
plt.show()
```

21.

3. Extract data of ‘rno, name, sub1, sub2, sub3, sub4, sub5’ directly from SQLite database and create dataframe using it (Use DataFrame()). And perform following tasks:

----extract data from sqlite and convert into dataframe----

```
import sqlite3
import pandas as pd
conn=sqlite3.connect('sales.db')
a=conn.execute('select * from data')
```

```
row=a.fetchall()
df=pd.DataFrame(row)
print(df)
```

3.1. Draw bar chart for Sub5 with respect to Name .

3.1.1. Give proper headings

```
import sqlite3
import pandas as pd import
matplotlib.pyplot as plt
conn=sqlite3.connect('salesmen.db'
)
a=conn.execute('select rno,name,s1,s2,s3,s4,s5 from
data') row=a.fetchall() df=pd.DataFrame(row)
print(df)
df= pd.DataFrame(row, columns=['rno', 'name', 's1', 's2', 's3', 's4', 's5'])
print(df) a=df['name'] b=df['s5'] plt.bar(a,b) plt.xlabel('name')
plt.ylabel('sub5') plt.title("student's marks") plt.show()
```

3.2. Draw scatter chart for Sub1, Sub2, Sub3, Sub4, Sub5 with different formatting.

3.2.1. Give proper headings.

```
import sqlite3
```

```
import pandas as pd import
matplotlib.pyplot as plt
conn=sqlite3.connect('salesmen.db'
)
a=conn.execute('select rno,name,s1,s2,s3,s4,s5 from
data') row=a.fetchall() df=pd.DataFrame(row)
print(df) df= pd.DataFrame(row, columns=['rno', 'name', 's1', 's2', 's3',
's4', 's5']) print(df) a=df['name'] s1=df['s1'] s2=df['s2'] s3=df['s3']
s4=df['s4'] s5=df['s5']
plt.scatter(a,s1,marker='o',c='red',s=100,edgecolor='black')
plt.scatter(a,s2,marker='^',c='yellow',s=200,edgecolor='black'
)
plt.scatter(a,s3,marker=',',c='green',s=100,edgecolor='black')
plt.scatter(a,s4,marker='<',c='blue',s=100,edgecolor='black')
plt.scatter(a,s5,marker='>',c='white',s=100,edgecolor='black')
plt.xlabel('name') plt.ylabel('marks') plt.title("student's marks")
plt.show()
```

3.3. Draw histogram chart for Sub4’s marks with proper heading.

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
conn=sqlite3.connect('salesmen.db')
a=conn.execute('select rno,name,s1,s2,s3,s4,s5 from
data') row=a.fetchall() df=pd.DataFrame(row)
print(df)
```

```
df= pd.DataFrame(row, columns=['rno', 'name', 's1', 's2', 's3', 's4', 's5'])
print(df) b=df['s4']
plt.hist(b,bins=5, color='skyblue', edgecolor='black')

plt.xlabel('marks')
plt.ylabel('frequency')
plt.title("student's marks")
plt.show()
```

3.4. Draw histogram chart for Sub2 as per [0,5,10,15,20,25,30,35,40,45,50] intervals.

```
import sqlite3
import pandas as pd import
matplotlib.pyplot as plt
conn=sqlite3.connect('salesmen.db'
)
a=conn.execute('select rno,name,s1,s2,s3,s4,s5 from
data') row=a.fetchall() df=pd.DataFrame(row)
print(df)
df= pd.DataFrame(row, columns=['rno', 'name', 's1', 's2', 's3', 's4', 's5'])
print(df)

b=df['s2']
plt.hist(b,bins=[0,5,10,15,20,25,30,35,40,45,50],
color='skyblue', edgecolor='black') plt.xlabel('marks')
plt.ylabel('frequency') plt.title("student's marks") plt.show()
```