

DBMS PRACTICAL- 8 PRELAB

1.What is Normalization?

DBMS Lab-8

190031249

Mohith

1. Normalization is the process of organizing the data in the database

→ Normalization is used to maximize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like insertion, update and deletion anomalies.

→ Normalization divides the larger tables into smaller table and links them using relationship.

2.What are various forms of Normalization?

2. Normalization :-

Here are the most commonly used normal forms:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce & Codd normal form (BCNF)

3. A table has fields F1, F2, F3, F4, F5 with the following functional dependencies
 $F1 \rightarrow F3$ $F2 \rightarrow F4$ $(F1 . F2) \rightarrow F5$

In terms of Normalization, this table is in which normal form?

3. INF (A relation is in first normal form if every attribute in that relation is single valued attribute.)

 Scanned with
CamScanner

4. What is a view?

4. View:

It is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

5. What is the difference between Clustered and Non-clustered index?

| 5. | Clustered Index | Non-clustered Index |
|----|---|---|
| | clustered index is faster. | Non-clustered index is slower. |
| | clustered index requires less memory for operations. | Non-clustered index requires more memory for operations. |
| | In this, index is the main data | In this, index is the copy of data |
| | A table can have only one clustered index. | A table can have multiple non-clustered index |
| | clustered index has inherent ability of storing data on the disk. | Non-clustered index store both value and a pointer to actual row that holds data. |

 Scanned with
CamScanner

6.Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values. F = {CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG} is a set of functional dependencies (FDs) so that F⁺ is exactly the set of FDs that hold for R.How many candidate keys does the relation R have

6. 4

Explanation

A⁺ is ABCDEFGH which is all attributes except D.

B⁺ is ABCEFGH which is all attributes except D.

E⁺ is also ABCEFGH which is all attributes except D.

F⁺ is also ABCEFGH which is all attributes except D.

so there are total 4 candidate keys

AD, BD, ED, FD

7.Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies { {E, F} \rightarrow {G}, {F} \rightarrow {I, J}, {E, H} \rightarrow {K, L}, K \rightarrow {M}, L \rightarrow {N} } on R. What is the key for R?

7. {E, F, H}

Explanation:-

All attributes can be derived from {E, F, H}

INLAB**Case Study 3 : WAREHOUSE SYSTEM****employee table:**

| eno | ename |
|-----|---------|
| 100 | Hari |
| 101 | Giri |
| 102 | Arun |
| 103 | Verma |
| 104 | Jaya |
| 105 | Kalyan |
| 106 | Krishna |
| 107 | Mohan |
| 108 | Bhasker |
| 109 | Arjun |

manager table:worker table:

| eno |
|-----|
| 102 |
| 103 |
| 105 |
| 107 |
| 109 |

| eno | mno |
|-----|-----|
| 101 | 102 |
| 104 | 103 |
| 106 | 102 |
| 108 | 105 |

phone table:

| eno | area_code | number |
|-----|-----------|------------|
| 100 | 91 | 9678897435 |
| 101 | 81 | 9743523134 |
| 102 | 61 | 9808148833 |
| 103 | 91 | 9872774532 |
| 104 | 91 | 9937400231 |
| 105 | 91 | 9675453111 |
| 106 | 81 | 9413505991 |
| 107 | 81 | 9151558871 |
| 108 | 91 | 8889611751 |
| 109 | 91 | 8627664631 |

address table:

| eno | street | city | state |
|-----|--------------|------------|----------------|
| 100 | Ameerpet | Hyderabad | Telangana |
| 101 | Raju Nagar | Guntur | Andhra Pradesh |
| 102 | Chowadavaram | Guntur | Andhra Pradesh |
| 103 | Kukatpally | Hyderabad | Telangana |
| 104 | BHEL | Hyderabad | Telangana |
| 105 | Poranki | Vijayawada | Andhra Pradesh |
| 106 | Bachupally | Hyderabad | Telangana |
| 107 | Nizampet | Hyderabad | Telangana |
| 108 | ECIL X Road | Hyderabad | Telangana |
| 109 | Benz Circle | Vijayawada | Andhra Pradesh |

warehouse table:**bin table:****part table:**

| partno |
|--------|
| 1000 |
| 1001 |
| 1002 |
| 1003 |
| 1004 |
| 1005 |
| 1006 |
| 1007 |
| 1008 |
| 1009 |

subpart table:

| partno | assmno |
|--------|--------|
| 1000 | 2001 |
| 1001 | 2000 |
| 1002 | 2001 |
| 1003 | 2000 |
| 1004 | 2000 |
| 1005 | 2002 |
| 1006 | 2002 |
| 1007 | 2001 |
| 1008 | 2000 |
| 1009 | 2002 |

item table:

| partno | itemno | batchno |
|--------|--------|---------|
| 1000 | 8000 | 7000 |
| 1005 | 8001 | 7002 |
| 1007 | 8002 | 7003 |
| 1008 | 8003 | 7004 |
| 1009 | 8004 | 7005 |

batch table:

| partno | batchno | size | date_in | mgrno | warehousid | binno |
|--------|---------|------|------------|-------|------------|-------|
| 1000 | 7000 | 10 | 13-03-2020 | 102 | 5000 | 6000 |

| | | | | | | |
|------|------|----|------------|-----|------|------|
| 1001 | 7000 | 20 | 25-02-2020 | 103 | 5000 | 6001 |
| 1002 | 7001 | 30 | 03-07-2020 | 105 | 5000 | 6002 |
| 1003 | 7002 | 40 | 15-04-2020 | 107 | 5001 | 6002 |
| 1004 | 7002 | 50 | 20-04-2020 | 109 | 5001 | 6003 |
| 1005 | 7003 | 60 | 07-08-2019 | 102 | 5002 | 6004 |
| 1006 | 7003 | 10 | 02-02-2020 | 105 | 5003 | 6001 |
| 1007 | 7004 | 20 | 10-02-2020 | 107 | 5003 | 6000 |
| 1008 | 7005 | 30 | 17-05-2020 | 105 | 5000 | 6003 |
| 1009 | 7005 | 30 | 20-07-2020 | 105 | 5004 | 6000 |

current_backorder table:

| partno | orgqty | remqty | bodate | backorder |
|--------|--------|--------|------------|-----------|
| 1000 | 1000 | 500 | 10-03-2020 | arun |
| 1005 | 500 | 300 | 15-02-2020 | kiran |
| 1007 | 700 | 400 | 12-03-2020 | raju |
| 1008 | 800 | 600 | 11-04-2020 | david |
| 1009 | 600 | 400 | 20-06-2020 | eswar |

old_backorder table:

| partno | orgqty | bodate | backorder | fulfilled |
|--------|--------|------------|-----------|-----------|
| 1000 | 1000 | 10-03-2020 | arun | yes |
| 1005 | 500 | 15-02-2020 | kiran | no |
| 1007 | 700 | 12-03-2020 | raju | yes |
| 1008 | 800 | 11-04-2020 | david | yes |
| 1009 | 600 | 20-06-2020 | eswar | yes |

- 1) Create all the tables required for creating the database for the Warehouse System case study.

```
create table employee(eno int primary key,ename varchar(10));
```

```
create table manager(eno int,foreign key(eno) references employee(eno));
```

```
create table worker(eno int,mno int,foreign key(eno) references employee(eno),foreign key(mno) references employee(eno));
```

```
create table address(eno int,street varchar(20),city varchar(20),state varchar(20),foreign key(eno) references employee(eno));
```

```
create table part(partno int primary key);
```

```
create table subpart(partno int,assmno int,foreign key(partno) references part(partno));
```

```
create table warehouse(warehouseid int primary key);
```

```
create table bin(warehouseid int,binno int primary key,capacity int,foreign key(warehouseid) references warehouse(warehouseid));
```

```
create table batch(partno int,batchno int,size int,date_in date,mgrno int,warehouseid int,binno int,foreign key(partno) references part(partno),foreign key(mgrno) references manager(eno),foreign key(warehouseid) references warehouse(warehouseid),foreign key(binno) references bin(binno));
```

```
create table current_backorder(partno int,orgqty int,remqty int,bodate date,backorder varchar(10),foreign key(partno) references part(partno));
```

```
create table old_backorder(partno int,orgqty int,bodate date,backorder varchar(10),fulfilled varchar(10),foreign key(partno) references part(partno));
```

```
create table phone(eno int,area_code int,number bigint,foreign key(eno) references employee(eno));
```

2) Insert atleast 10 records into all the tables

Employee Data Insertion:

```

18 • insert into employee values(104,'Jaya');
19 • insert into employee values(105,'Kalyan');
20 • insert into employee values(106,'Krishna');
21 • insert into employee values(107,'Mohan');
22 • insert into employee values(108,'Bhasker');
23 • insert into employee values(109,'Arjun');
24 • select * from employee;
25

```

| eno | ename |
|-----|---------|
| 100 | Hari |
| 101 | Giri |
| 102 | Arun |
| 103 | Verma |
| 104 | Jaya |
| 105 | Kalyan |
| 106 | Krishna |

Manager Data Insertion:

```

26 • insert into manager values(102);
27 • insert into manager values(103);
28 • insert into manager values(105);
29 • insert into manager values(107);
30 • insert into manager values(109);
31 • select * from manager;
32
33 • insert into worker values(101,102);

```

| eno |
|-----|
| 102 |
| 103 |
| 105 |
| 107 |
| 109 |

SQL File 3*

```

32
33 • insert into worker values(101,102);
34 • insert into worker values(104,103);
35 • insert into worker values(106,102);
36 • insert into worker values(108,105);
37 • select *from workers;
38
39 • insert into address values(100,'Ameerpet','Hyderabad','Telangana');

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor

| eno | mno |
|-----|-----|
| 101 | 102 |
| 104 | 103 |
| 106 | 102 |
| 108 | 105 |

SQL File 3*

```

43 • insert into address values(104,'BHEL','Hyderabad','Telangana');
44 • insert into address values(105,'Poranki','Vijayawada','Andhra Pradesh');
45 • insert into address values(106,'Bachupally','Hyderabad','Telangana');
46 • insert into address values(107,'Nizampet','Hyderabad','Telangana');
47 • insert into address values(108,'ECIL X Road','Hyderabad','Telangana');
48 • insert into address values(109,'Benz Circle','Vijayawada','Andhra Pradesh');
49 • select *from address;
50

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Form Editor Read Only

| eno | street | city | state |
|-----|-------------|------------|----------------|
| 100 | Ameerpet | Hyderabad | Telangana |
| 101 | Raju Nagar | Guntur | Andhra Pradesh |
| 102 | Chowdavaram | Guntur | Andhra Pradesh |
| 103 | Kukatpally | Hyderabad | Telangana |
| 104 | BHEL | Hyderabad | Telangana |
| 105 | Poranki | Vijayawada | Andhra Pradesh |
| 106 | Bachupally | Hyderabad | Telangana |

SQL File 3*

```

55 • insert into part values(1004);
56 • insert into part values(1005);
57 • insert into part values(1006);
58 • insert into part values(1007);
59 • insert into part values(1008);
60 • insert into part values(1009);
61 • select *from part;
62

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid Form Editor Apply

| partno |
|--------|
| 1000 |
| 1001 |
| 1002 |
| 1003 |
| 1004 |
| 1005 |
| 1006 |

SQL File 3* x

```

67 • insert into subpart values(1004,2000);
68 • insert into subpart values(1005,2002);
69 • insert into subpart values(1006,2002);
70 • insert into subpart values(1007,2001);
71 • insert into subpart values(1008,2000);
72 • insert into subpart values(1009,2002);
73 • select *from subpart;
74

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Read Only

| partno | assmno |
|--------|--------|
| 1000 | 2001 |
| 1001 | 2000 |
| 1002 | 2001 |
| 1003 | 2000 |
| 1004 | 2000 |
| 1005 | 2002 |
| 1006 | 2002 |

SQL File 3* x

```

74
75 • insert into warehouse values(5000);
76 • insert into warehouse values(5001);
77 • insert into warehouse values(5002);
78 • insert into warehouse values(5003);
79 • insert into warehouse values(5004);
80 • insert into warehouse values(5005);
81 • select *from warehouse;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Apply

| warehouseid |
|-------------|
| 5000 |
| 5001 |
| 5002 |
| 5003 |
| 5004 |
| 5005 |
| * NULL |

SQL File 3* x

```

82
83 • insert into bin values(5000,6000,100);
84 • insert into bin values(5001,6001,150);
85 • insert into bin values(5002,6002,200);
86 • insert into bin values(5003,6003,250);
87 • insert into bin values(5004,6004,300);
88 • insert into bin values(5005,6005,200);
89 • select *from bin;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Apply

| warehouseid | binno | capacity |
|-------------|--------|----------|
| 5000 | 6000 | 100 |
| 5001 | 6001 | 150 |
| 5002 | 6002 | 200 |
| 5003 | 6003 | 250 |
| 5004 | 6004 | 300 |
| 5005 | 6005 | 200 |
| * NULL | * NULL | * NULL |

SQL File 3* ×

```

95 • insert into batch values(1004,7002,50,str_to_date('20-04-2020','%d-%m-%Y'),109,5001,6003);
96 • insert into batch values(1005,7003,60,str_to_date('07-08-2019','%d-%m-%Y'),102,5002,6004);
97 • insert into batch values(1006,7003,10,str_to_date('02-02-2020','%d-%m-%Y'),105,5003,6001);
98 • insert into batch values(1007,7004,20,str_to_date('10-02-2020','%d-%m-%Y'),107,5003,6000);
99 • insert into batch values(1008,7005,30,str_to_date('17-05-2020','%d-%m-%Y'),105,5000,6003);
100 • insert into batch values(1009,7005,30,str_to_date('20-07-2020','%d-%m-%Y'),105,5004,6000);
101 • select *from batch;
102

```

Result Grid | Filter Rows: [] Export: [] Wrap Cell Content: []

| partno | batchno | size | date_in | mgrno | warehouseid | binno |
|--------|---------|------|------------|-------|-------------|-------|
| 1000 | 7000 | 10 | 2020-03-13 | 102 | 5000 | 6000 |
| 1001 | 7000 | 20 | 2020-02-25 | 103 | 5000 | 6001 |
| 1002 | 7001 | 30 | 2020-07-03 | 105 | 5000 | 6002 |
| 1003 | 7002 | 40 | 2020-04-15 | 107 | 5001 | 6002 |
| 1004 | 7002 | 50 | 2020-04-20 | 109 | 5001 | 6003 |
| 1005 | 7003 | 60 | 2019-08-07 | 102 | 5002 | 6004 |
| 1006 | 7003 | 10 | 2020-02-07 | 105 | 5003 | 6001 |

batch 39 ×

SQL File 3* ×

```

103 • insert into current_backorder values(1000,1000,500,str_to_date('10-03-2020','%d-%m-%Y'),'arun');
104 • insert into current_backorder values(1005,500,300,str_to_date('15-02-2020','%d-%m-%Y'),'kiran');
105 • insert into current_backorder values(1007,700,400,str_to_date('12-03-2020','%d-%m-%Y'),'raju');
106 • insert into current_backorder values(1008,800,600,str_to_date('11-04-2020','%d-%m-%Y'),'david');
107 • insert into current_backorder values(1009,600,400,str_to_date('20-06-2020','%d-%m-%Y'),'eswar');
108 • select *from current_backorder;
109
110 • insert into old_backorder values(1000,1000,str_to_date('10-03-2020','%d-%m-%Y'),'arun','yes');

```

Result Grid | Filter Rows: [] Export: [] Wrap Cell Content: []

| partno | orgqty | remqty | bodate | backorder |
|--------|--------|--------|------------|-----------|
| 1000 | 1000 | 500 | 2020-03-10 | arun |
| 1005 | 500 | 300 | 2020-02-15 | kiran |
| 1007 | 700 | 400 | 2020-03-12 | raju |
| 1008 | 800 | 600 | 2020-04-11 | david |
| 1009 | 600 | 400 | 2020-06-20 | eswar |

Toggle wrapping of cell contents

SQL File 3* ×

```

109
110 • insert into old_backorder values(1000,1000,str_to_date('10-03-2020','%d-%m-%Y'),'arun','yes');
111 • insert into old_backorder values(1005,500,str_to_date('15-02-2020','%d-%m-%Y'),'kiran','no');
112 • insert into old_backorder values(1007,700,str_to_date('12-03-2020','%d-%m-%Y'),'raju','yes');
113 • insert into old_backorder values(1008,800,str_to_date('11-04-2020','%d-%m-%Y'),'david','yes');
114 • insert into old_backorder values(1009,600,str_to_date('20-06-2020','%d-%m-%Y'),'eswar','yes');
115 • select *from old_backorder;
116

```

Result Grid | Filter Rows: [] Export: [] Wrap Cell Content: []

| partno | orgqty | bodate | backorder | fulfilled |
|--------|--------|------------|-----------|-----------|
| 1000 | 1000 | 2020-03-10 | arun | yes |
| 1005 | 500 | 2020-02-15 | kiran | no |
| 1007 | 700 | 2020-03-12 | raju | yes |
| 1008 | 800 | 2020-04-11 | david | yes |
| 1009 | 600 | 2020-06-20 | eswar | yes |

```

122 • insert into phone values(105,91,9675453111);
123 • insert into phone values(106,81,9413505991);
124 • insert into phone values(107,81,9151558871);
125 • insert into phone values(108,91,8889611751);
126 • insert into phone values(109,91,8627664631);
127 • select *from phone;
128
129 -- 2

```

| eno | area_code | number |
|-----|-----------|------------|
| 100 | 91 | 9678897435 |
| 101 | 81 | 9743523134 |
| 102 | 61 | 9808148833 |
| 103 | 91 | 9872774532 |
| 104 | 91 | 9937400231 |
| 105 | 91 | 9675453111 |
| 106 | 81 | 9413505991 |

- 3) Display all employee numbers for all the workers that work under manager with the name starting with "S"

select w.eno from worker w,employee e where w.eno = e.eno and e.ename like 'S%';

```

124 • insert into phone values(107,81,9151558871);
125 • insert into phone values(108,91,8889611751);
126 • insert into phone values(109,91,8627664631);
127 • select *from phone;
128
129 -- 3
130 • select w.eno from worker w,employee e where w.eno = e.eno and e.ename like 'S%';
131 -- 3

```

| eno |
|-----|
| 108 |
| 109 |
| 107 |

- 4) Display name and employee number of all employees in the ascending order on their name. The names should be listed in an alphabetic order

select *from employee order by ename asc;

SQL File 3*

```

127 • select *from phone;
128
129 -- 3
130 • select w.eno from worker w,employee e where w.eno = e.eno and e.ename like 'S%';
131 -- 4
132 • select *from employee order by ename asc;
133 -- 5
134 • select p.* from phone p,manager m where p.eno = m.eno;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Apply

| eno | ename |
|-----|---------|
| 109 | Arjun |
| 102 | Arun |
| 108 | Bhasker |
| 101 | Giri |
| 100 | Hari |
| 104 | Jaya |
| 105 | Kalvan |

employee 46 x

5) Display all the phones and employee number for all the managers.

SQL File 3*

```

128
129 -- 3
130 • select w.eno from worker w,employee e where w.eno = e.eno and e.ename like 'S%';
131 -- 4
132 • select *from employee order by ename asc;
133 -- 5
134 • select p.* from phone p,manager m where p.eno = m.eno;
135 -- 6

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Read Only

| eno | area_code | number |
|-----|-----------|------------|
| 102 | 61 | 9808148833 |
| 103 | 91 | 9872774532 |
| 105 | 91 | 9675453111 |
| 107 | 81 | 9151558871 |
| 109 | 91 | 8627664631 |

Result 47 x

6) Display the all parts that are assemblies they should be listed in ascending order.

SQL File 3*

```

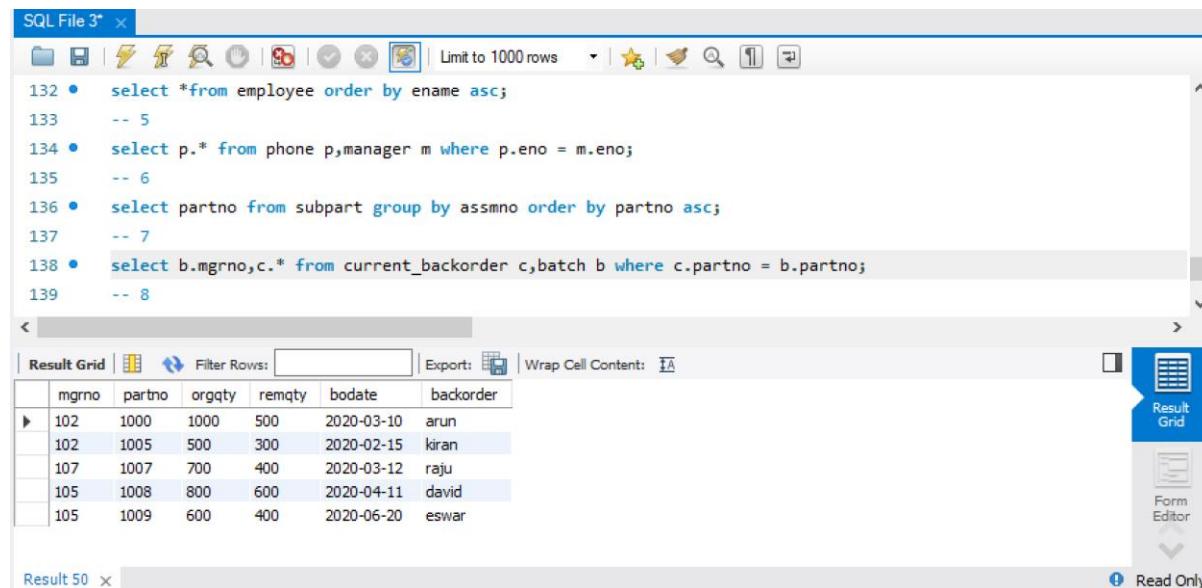
130 • select w.eno from worker w,employee e where w.eno = e.eno and e.ename like 'S%';
131 -- 4
132 • select *from employee order by ename asc;
133 -- 5
134 • select p.* from phone p,manager m where p.eno = m.eno;
135 -- 6
136 • select partno from subpart group by assmno order by partno asc;
137 -- 7

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form

| partno |
|--------|
| 1000 |
| 1001 |
| 1005 |

- 7) Display all current backorders done by each manager.



The screenshot shows the SQL Developer interface with a query editor and a result grid. The query is:

```

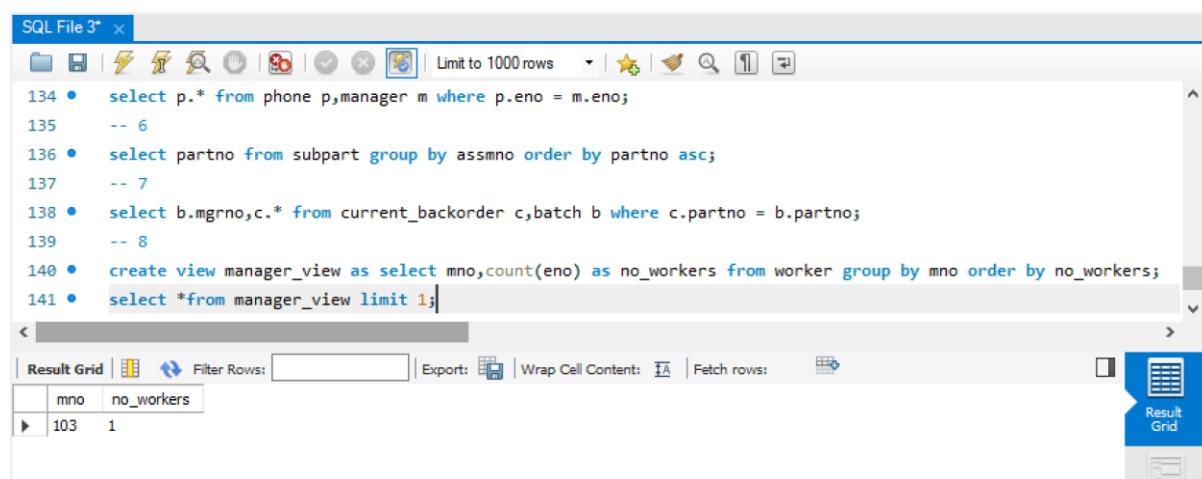
132 • select *from employee order by ename asc;
133 -- 5
134 • select p.* from phone p,manager m where p.eno = m.eno;
135 -- 6
136 • select partno from subpart group by assmno order by partno asc;
137 -- 7
138 • select b.mgrno,c.* from current_backorder c,batch b where c.partno = b.partno;
139 -- 8

```

The result grid displays the following data:

| mgrno | partno | orgqty | reqqty | bodate | backorder |
|-------|--------|--------|--------|------------|-----------|
| 102 | 1000 | 1000 | 500 | 2020-03-10 | arun |
| 102 | 1005 | 500 | 300 | 2020-02-15 | kiran |
| 107 | 1007 | 700 | 400 | 2020-03-12 | raju |
| 105 | 1008 | 800 | 600 | 2020-04-11 | david |
| 105 | 1009 | 600 | 400 | 2020-06-20 | eswar |

- 8) Display employee id and number of workers managed for all the managers with the smallest number of workers managed.



The screenshot shows the SQL Developer interface with a query editor and a result grid. The query is:

```

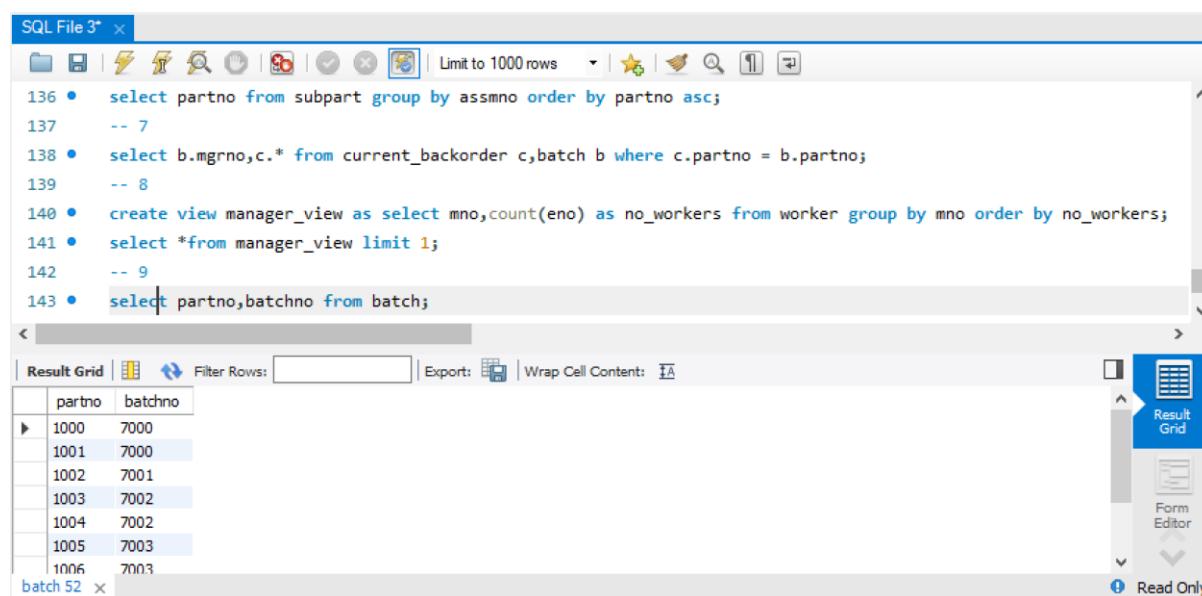
134 • select p.* from phone p,manager m where p.eno = m.eno;
135 -- 6
136 • select partno from subpart group by assmno order by partno asc;
137 -- 7
138 • select b.mgrno,c.* from current_backorder c,batch b where c.partno = b.partno;
139 -- 8
140 • create view manager_view as select mno,count(eno) as no_workers from worker group by mno order by no_workers;
141 • select *from manager_view limit 1;

```

The result grid displays the following data:

| mno | no_workers |
|-----|------------|
| 103 | 1 |

- 9) Display the list of all parts with their batchno



The screenshot shows the SQL Developer interface with a query editor and a result grid. The query is:

```

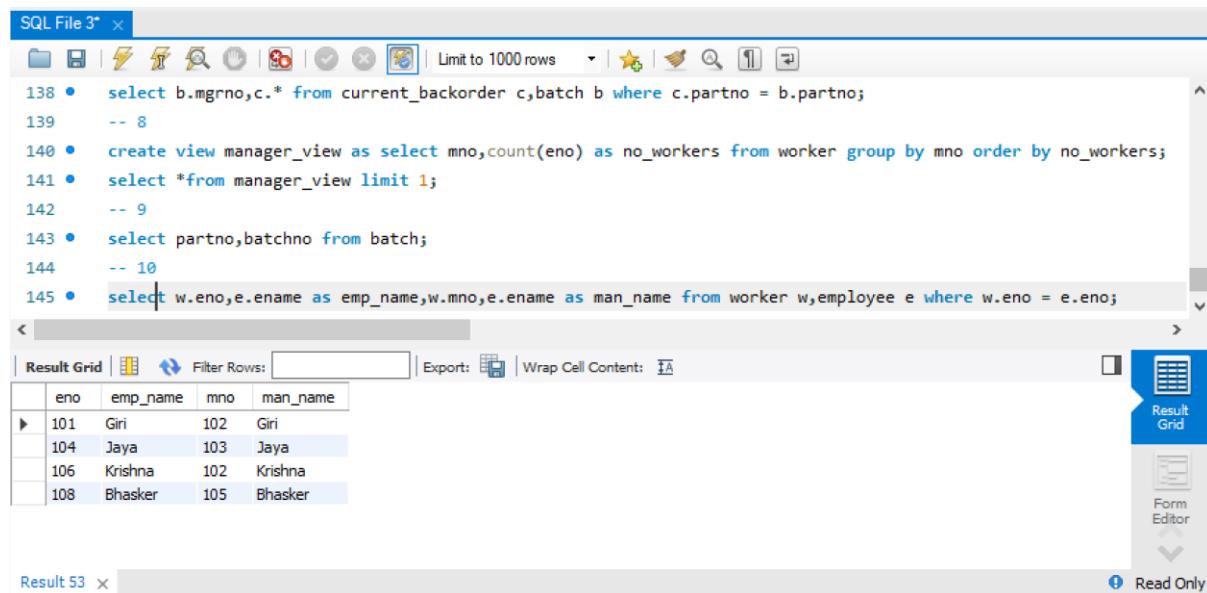
136 • select partno from subpart group by assmno order by partno asc;
137 -- 7
138 • select b.mgrno,c.* from current_backorder c,batch b where c.partno = b.partno;
139 -- 8
140 • create view manager_view as select mno,count(eno) as no_workers from worker group by mno order by no_workers;
141 • select *from manager_view limit 1;
142 -- 9
143 • select partno,batchno from batch;

```

The result grid displays the following data:

| partno | batchno |
|--------|---------|
| 1000 | 7000 |
| 1001 | 7000 |
| 1002 | 7001 |
| 1003 | 7002 |
| 1004 | 7002 |
| 1005 | 7003 |
| 1006 | 7003 |

10) Display the employee details with their manager details



```

SQL File 3* x
138 •   select b.mgrno,c.* from current_backorder c,batch b where c.partno = b.partno;
139     -- 8
140 •   create view manager_view as select mno,count(eno) as no_workers from worker group by mno order by no_workers;
141 •   select *from manager_view limit 1;
142     -- 9
143 •   select partno,batchno from batch;
144     -- 10
145 •   select w.eno,e.ename as emp_name,w.mno,e.ename as man_name from worker w,employee e where w.eno = e.eno;

```

The screenshot shows the SQL developer interface with the following details:

- Result Grid:** Shows the output of the query. The columns are eno, emp_name, mno, and man_name. The data is as follows:

| eno | emp_name | mno | man_name |
|-----|----------|-----|----------|
| 101 | Giri | 102 | Giri |
| 104 | Jaya | 103 | Jaya |
| 106 | Krishna | 102 | Krishna |
| 108 | Bhasker | 105 | Bhasker |
- Result Grid Options:** Includes buttons for Result Grid, Form Editor, and Read Only mode.
- Result 53:** Shows the total number of rows returned.

Case Study 6 : PAINTING HIRE BUSINESS

CUSTOMER:

| cid | cname | address | phone | category |
|-----|---------|------------|------------|----------|
| 100 | Raju | Hyderabad | 9876045789 | Bronze |
| 101 | Hari | Vijayawada | 8877678956 | Gold |
| 102 | Devi | Guntur | 7879312123 | Silver |
| 103 | Rani | Delhi | 8780945290 | Platinum |
| 104 | Jaya | Mumbai | 9612578457 | Gold |
| 105 | Haritha | Kolkata | 9611665513 | Silver |
| 106 | Kalyan | Vijayawada | 9610752569 | Bronze |
| 107 | Roja | Hyderabad | 9609839625 | Platinum |
| 108 | Amar | Vijayawada | 9608926681 | Gold |
| 109 | Padma | Vijayawada | 9608013737 | Bronze |

ARTIST:

| aid | name | address | phone |
|-----|--------|-----------|------------|
| 200 | John | Delhi | 7786549803 |
| 201 | Samuel | Mumbai | 7123458790 |
| 202 | Samson | Lucknow | 9460367777 |
| 203 | David | Hyderabad | 9797276764 |
| 204 | Raghu | Hyderabad | 8134185751 |
| 205 | Ravi | Mumbai | 7471094738 |
| 206 | Kiran | Delhi | 9808003725 |

PAINTING:

| pid | aid | rental_cost | type |
|-----|-----|-------------|-----------|
| 300 | 201 | 4500 | Hired |
| 301 | 202 | 3500 | Not hired |
| 302 | 203 | 7500 | Hired |
| 303 | 205 | 2500 | Not hired |
| 304 | 202 | 10000 | Not hired |
| 305 | 201 | 8000 | Not hired |
| 306 | 203 | 6500 | Not hired |

RENT:

| cid | pid | renatl_date | rental_period |
|-----|-----|-------------|---------------|
| 104 | 300 | 10-06-2020 | 6 |
| 105 | 302 | 05-07-2020 | 10 |
| 108 | 304 | 15-07-2020 | 3 |
| 109 | 305 | 25-06-2020 | 6 |

OWNER:

| pid | oid | name | address | phone |
|-----|-----|---------|------------|------------|
| 300 | 500 | Raju | Hyderabad | 9460367777 |
| 301 | 500 | Hari | Vijayawada | 8134185751 |
| 302 | 501 | Giri | Hyderabad | 7808003725 |
| 303 | 501 | Gopi | Delhi | 9481821699 |
| 304 | 503 | Krishna | Mumbai | 7155639673 |
| 305 | 502 | Verma | Delhi | 8829457647 |
| 306 | 502 | Guna | Delhi | 7503275621 |

- 1) Create the tables identifying the constraints and relationships

```
create table Customer(cid int primary key,cname varchar(10),address varchar(10),phone
bigint,category varchar(10));
```

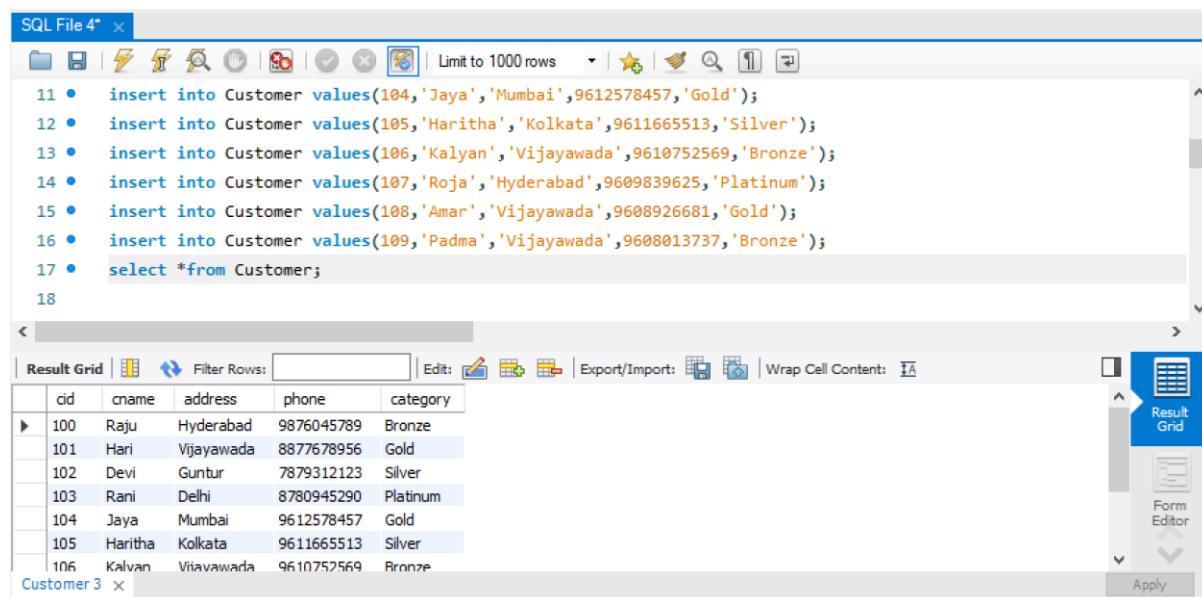
```
create table Artist(aid int primary key,name varchar(10),address varchar(10),phone bigint);
```

```
create table Painting(pid int primary key,aid int,rental_cost int,type varchar(10),foreign key(aid)
references Artist(aid));
```

```
create table Owner(pid int,oid int,name varchar(10),address varchar(10),phone bigint,foreign key(pid)
references Painting(pid));
```

```
create table Rent(cid int,pid int,rental_date date,rental_period int,foreign key(cid) references
Customer(cid),foreign key(pid) references Painting(pid));
```

- 2) Insert atleast 10 records into these tables



The screenshot shows the MySQL Workbench interface. The top part is the SQL Editor titled "SQL File 4*". It contains the following SQL statements:

```

11 • insert into Customer values(104,'Jaya','Mumbai',9612578457,'Gold');
12 • insert into Customer values(105,'Haritha','Kolkata',9611665513,'Silver');
13 • insert into Customer values(106,'Kalyan','Vijayawada',9610752569,'Bronze');
14 • insert into Customer values(107,'Roja','Hyderabad',9609839625,'Platinum');
15 • insert into Customer values(108,'Amar','Vijayawada',9608926681,'Gold');
16 • insert into Customer values(109,'Padma','Vijayawada',9608013737,'Bronze');
17 • select *from Customer;
18

```

The bottom part is the "Result Grid" window, which displays the following data:

| cid | cname | address | phone | category |
|-----|---------|------------|------------|----------|
| 100 | Raju | Hyderabad | 9876045789 | Bronze |
| 101 | Hari | Vijayawada | 8877678956 | Gold |
| 102 | Devi | Guntur | 7879312123 | Silver |
| 103 | Rani | Delhi | 8780945290 | Platinum |
| 104 | Jaya | Mumbai | 9612578457 | Gold |
| 105 | Haritha | Kolkata | 9611665513 | Silver |
| 106 | Kalvan | Vijayawada | 9610752569 | Bronze |

SQL File 4*

```

20 • insert into Artist values(201,'Samuel','Mumbai',7123458790);
21 • insert into Artist values(202,'Samson','Lucknow',9460367777);
22 • insert into Artist values(203,'David','Hyderabad',9797276764);
23 • insert into Artist values(204,'Raghu','Hyderabad',8134185751);
24 • insert into Artist values(205,'Ravi','Mumbai',7471094738);
25 • insert into Artist values(206,'Kiran','Delhi',9808003725);
26 • select *from Artist;
27 • insert into Painting values (300,201,4500,'Hired');

```

Result Grid

| aid | name | address | phone |
|-----|--------|-----------|------------|
| 200 | John | Delhi | 7786549803 |
| 201 | Samuel | Mumbai | 7123458790 |
| 202 | Samson | Lucknow | 9460367777 |
| 203 | David | Hyderabad | 9797276764 |
| 204 | Raghu | Hyderabad | 8134185751 |
| 205 | Ravi | Mumbai | 7471094738 |
| 206 | Kiran | Delhi | 9808003725 |

Artist 4

SQL File 4*

```

29 • insert into Painting values (302,203,7500,'Hired');
30 • insert into Painting values (303,205,2500,'Not hired');
31 • insert into Painting values (304,202,10000,'Not hired');
32 • insert into Painting values (305,201,8000,'Not hired');
33 • insert into Painting values (306,203,6500,'Not hired');
34 • select *from Painting;
35 • insert into Owner values(300,500,'Raju','Hyderabad',9460367777);
36 • insert into Owner values(301,500,'Hari','Vijayawada',8134185751);

```

Result Grid

| pid | aid | rental_cost | type |
|-----|-----|-------------|-----------|
| 300 | 201 | 4500 | Hired |
| 301 | 202 | 3500 | Not hired |
| 302 | 203 | 7500 | Hired |
| 303 | 205 | 2500 | Not hired |
| 304 | 202 | 10000 | Not hired |
| 305 | 201 | 8000 | Not hired |
| 306 | 203 | 6500 | Not hired |

Painting 5

SQL File 4*

```

35 • insert into Owner values(300,500,'Raju','Hyderabad',9460367777);
36 • insert into Owner values(301,500,'Hari','Vijayawada',8134185751);
37 • insert into Owner values(302,501,'Giri','Hyderabad',7808003725);
38 • insert into Owner values(303,501,'Gopi','Delhi',9481821699);
39 • insert into Owner values(304,503,'Krishna','Mumbai',7155639673);
40 • insert into Owner values(305,502,'Verma','Delhi',8829457647);
41 • insert into Owner values(306,502,'Guna','Delhi',7503275621);
42 • select *from Owner;

```

Result Grid

| pid | oid | name | address | phone |
|-----|-----|---------|------------|------------|
| 300 | 500 | Raju | Hyderabad | 9460367777 |
| 301 | 500 | Hari | Vijayawada | 8134185751 |
| 302 | 501 | Giri | Hyderabad | 7808003725 |
| 303 | 501 | Gopi | Delhi | 9481821699 |
| 304 | 503 | Krishna | Mumbai | 7155639673 |
| 305 | 502 | Verma | Delhi | 8829457647 |

Owner 6

Read Only

SQL File 4*

```

40 • insert into Owner values(305,502,'Verma','Delhi',8829457647);
41 • insert into Owner values(306,502,'Guna','Delhi',7503275621);
42 • select *from Owner;
43 • insert into Rent values(104,300,str_to_date('10-06-2020','%d-%m-%Y'),6);
44 • insert into Rent values(105,302,str_to_date('05-07-2020','%d-%m-%Y'),10);
45 • insert into Rent values(108,304,str_to_date('15-07-2020','%d-%m-%Y'),3);
46 • insert into Rent values(109,305,str_to_date('25-06-2020','%d-%m-%Y'),6);
47 • select *from Rent;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor

| cid | pid | rental_date | rental_period |
|-----|-----|-------------|---------------|
| 104 | 300 | 2020-06-10 | 6 |
| 105 | 302 | 2020-07-05 | 10 |
| 108 | 304 | 2020-07-15 | 3 |
| 109 | 305 | 2020-06-25 | 6 |

3) Display the customer details who got the category as "Gold"

SQL File 4*

```

42 • select *from Owner;
43 • insert into Rent values(104,300,str_to_date('10-06-2020','%d-%m-%Y'),6);
44 • insert into Rent values(105,302,str_to_date('05-07-2020','%d-%m-%Y'),10);
45 • insert into Rent values(108,304,str_to_date('15-07-2020','%d-%m-%Y'),3);
46 • insert into Rent values(109,305,str_to_date('25-06-2020','%d-%m-%Y'),6);
47 • select *from Rent;
48 -- 3
49 • select *from Customer where category = 'Gold';

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor

| cid | cname | address | phone | category |
|-----|-------|------------|------------|----------|
| 101 | Hari | Vijayawada | 8877678956 | Gold |
| 104 | Jaya | Mumbai | 9612578457 | Gold |
| 108 | Amar | Vijayawada | 9608926681 | Gold |
| * | NULL | NULL | NULL | NULL |

4) Display the list of artists who belong to "Hyderabad"

SQL File 4*

```

44 • insert into Rent values(105,302,str_to_date('05-07-2020','%d-%m-%Y'),10);
45 • insert into Rent values(108,304,str_to_date('15-07-2020','%d-%m-%Y'),3);
46 • insert into Rent values(109,305,str_to_date('25-06-2020','%d-%m-%Y'),6);
47 • select *from Rent;
48 -- 3
49 • select *from Customer where category = 'Gold';
50 -- 4
51 • select *from Artist where address = 'Hyderabad';

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form

| aid | name | address | phone |
|-----|-------|-----------|------------|
| 203 | David | Hyderabad | 9797276764 |
| 204 | Raghu | Hyderabad | 8134185751 |
| * | NULL | NULL | NULL |

5) Create a query display the painting details which are not hired

SQL File 4*

```

46 • insert into Rent values(109,305,str_to_date('25-06-2020','%d-%m-%Y'),6);
47 • select *from Rent;
48 -- 3
49 • select *from Customer where category = 'Gold';
50 -- 4
51 • select *from Artist where address = 'Hyderabad';
52 -- 5
53 • select *from Painting where type = "Not hired";

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Apply

| pid | aid | rental_cost | type |
|-----|------|-------------|-----------|
| 301 | 202 | 3500 | Not hired |
| 303 | 205 | 2500 | Not hired |
| 304 | 202 | 10000 | Not hired |
| 305 | 201 | 8000 | Not hired |
| 306 | 203 | 6500 | Not hired |
| * | HULL | HULL | HULL |

Painting 11 *

- 6) Create a query to update the rental cost of the painting with 1000 for the paintings which are not hired

SQL File 4*

```

49 • select *from Customer where category = 'Gold';
50 -- 4
51 • select *from Artist where address = 'Hyderabad';
52 -- 5
53 • select *from Painting where type = "Not hired";
54 -- 6
55 • update Painting set rental_cost = 1000 where type = "Not hired";
56 • select *from Painting;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Result Grid | Form Editor | Apply

| pid | aid | rental_cost | type |
|-----|------|-------------|-----------|
| 300 | 201 | 4500 | Hired |
| 301 | 202 | 1000 | Not hired |
| 302 | 203 | 7500 | Hired |
| 303 | 205 | 1000 | Not hired |
| 304 | 202 | 1000 | Not hired |
| 305 | 201 | 1000 | Not hired |
| 306 | 203 | 1000 | Not hired |
| * | HULL | HULL | HULL |

Painting 15 *

- 7) Display the details of the artist whose paintings are hired

SQL File 4*

```

51 • select *from Artist where address = 'Hyderabad';
52 -- 5
53 • select *from Painting where type = "Not hired";
54 -- 6
55 • update Painting set rental_cost = 1000 where type = "Not hired";
56 • select *from Painting;
57 -- 7
58 • select a.* from Artist a,Painting p where a.aid = p.aid and p.type = "Hired";

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid | Form Editor | Apply

| aid | name | address | phone |
|-----|--------|-----------|------------|
| 201 | Samuel | Mumbai | 7123458790 |
| 203 | David | Hyderabad | 9797276764 |

POST-LAB

- 1) Company 'ABC' has decided to get the details of every employee. The company wants to know *id, name, manager id, manager name and computer brand* of every employee. Your job is to generate the report in the above mentioned format. Output must be in ascending order of employee name.

Table: computer

| Field | Type |
|-------|---------|
| id | int |
| brand | varchar |

Table:Employee**Table: employee**

| Field | Type |
|---------|------|
| id | int |
| name | text |
| manager | int |
| comp_id | int |

Table: Computer

| id | brand |
|------|---------|
| 1001 | Dell |
| 1002 | HP |
| 1003 | Lenovo |
| 1004 | Asus |
| 1005 | Compac |
| 1006 | Apple |
| 1007 | Samsung |
| 1008 | Acer |
| 1009 | Sony |

| id | name | manager | comp_id |
|--------|----------|---------|---------|
| 100001 | Vishal | 100002 | 1004 |
| 100002 | Kanak | 100005 | 1006 |
| 100003 | Dinesh | 100001 | 1002 |
| 100004 | Rajesh | 100003 | 1001 |
| 100005 | Rohan | 100001 | 1007 |
| 100006 | Manisha | 100004 | 1008 |
| 100007 | Prabhat | 100005 | 1006 |
| 100008 | Abhishek | 100007 | 1006 |

Table: Output

| id | name | manager_id | manager_name | comp_brand |
|--------|----------|------------|--------------|------------|
| 100008 | Abhishek | 100007 | Prabhat | Apple |
| 100003 | Dinesh | 100001 | Vishal | HP |
| 100002 | Kanak | 100005 | Rohan | Apple |

| | | | | |
|--------|---------|--------|--------|---------|
| 100006 | Manisha | 100004 | Rajesh | Acer |
| 100007 | Prabhat | 100005 | Rohan | Apple |
| 100004 | Rajesh | 100003 | Dinesh | Dell |
| 100005 | Rohan | 100001 | Vishal | Samsung |
| 100001 | Vishal | 100002 | Kanak | Asus |

ANS)

```
select e.id,e.name,e.manager as manager_id,e1.name as
manager_name,c.brand as comp_brand from Employee e join Computer c
on e.comp_id = c.id join Employee e1 on e1.id = e.manager order by e.name
asc;
```

```
14 • create table Employee(id int primary key,name varchar(10),manager int,comp_id int,foreign key(comp_id) references Computer(id));
15 • insert into Employee values(100001,'Vishal',100002,1004);
16 • insert into Employee values(100002,'Kanak',100005,1005);
17 • insert into Employee values(100003,'Dinesh',100001,1002);
18 • insert into Employee values(100004,'Rajesh',100003,1001);
19 • insert into Employee values(100005,'Rohan',100001,1007);
20 • insert into Employee values(100006,'Manisha',100004,1008);
21 • insert into Employee values(100007,'Prabhat',100005,1006);
22 • insert into Employee values(100008,'Abhishek',100007,1006);
23 • select *from Employee;
24 • select e.id,e.name,e.manager as manager_id,e1.name as manager_name,c.brand as comp_brand from Employee e join Computer c on e.comp_id = c.id join Employee e1 on e1.id = e.manager order by e.name asc;
```

| id | name | manager_id | manager_name | comp_brand |
|--------|----------|------------|--------------|------------|
| 100001 | Abhishek | 100007 | Prabhat | Apple |
| 100003 | Dinesh | 100005 | Vishal | HP |
| 100004 | Manisha | 100006 | Rohan | Apple |
| 100006 | Rajesh | 100004 | Rajesh | Acer |
| 100007 | Prabhat | 100005 | Rohan | Apple |
| 100008 | Rajesh | 100003 | Dinesh | Dell |
| 100005 | Rohan | 100001 | Vishal | Samsung |
| 100001 | Vishal | 100002 | Kanak | Asus |

2) Micro has made a new search algorithm. What it does is check whether the query string is a subsequence of the search string or not. A string *A* is considered a subsequence of string *B* if some characters from *B* can be deleted so that it becomes equal to *A*. Micro ran his algorithm on a list of messages he has and got some output but now he needs to verify it. Help Micro verify the output by doing a search on the same list of messages. String he queried for is "hack" (without quotes).

Input Format:

Table : **Messages**

| Field | Type |
|---------|---------|
| Content | text |
| id | integer |

Sample:

| Content | id |
|---------|----|
| hacker | 1 |
| hacak | 2 |
| happy | 3 |

Output:

| Content | id |
|---------|----|
| hacker | 1 |
| hacak | 2 |

Explanation

String "hack" appears as a subsequence in strings "hacker" and "hacak".

ANS) **select *from Messages where Content like '%h%a%c%k%' order by id asc;**

3) You are given a table Employee.Employee: It will consist of 2 column, first is Employee Id, and second is the Joining Date of that employee.Date format will be: YYYY-MM-DD. You need to find the number of joinings happened on the even days, for all the special years in the given data.Special years are the ones, in which joining happened only in the even months in the given data.

Example:

i) Even months of an year are : February (2nd month), April (4th month) and so on.

ii) Number of even days of a normal year are: Given dates: 1999-12-01 (odd day), 1999-02-02 (even day), 1999-06-08 (even day). Here number of joinings happened on even days for year 1999 is 2.

Note: Output the data in the ascending order of the Special Year.

Input Format:

Output Format:

Table : Employee

| Field | Type |
|-------------|------|
| EmployeeID | int |
| JoiningDate | Date |

| Field | Type |
|--------------|------|
| Special_Year | int |
| EvenDays | int |

Sample Input:

Employee Table:

| EmployeeID | JoiningDate |
|------------|-------------|
| 1 | 1998-12-01 |
| 2 | 1998-06-02 |
| 3 | 1998-02-28 |
| 4 | 2000-04-16 |
| 5 | 2000-04-18 |
| 6 | 2000-06-18 |
| 7 | 1999-08-12 |
| 8 | 1999-10-14 |
| 9 | 1999-10-15 |
| 10 | 2002-01-12 |
| 11 | 2002-02-14 |

Sample Output:

| Special_Year | EvenDays |
|--------------|----------|
| 1998 | 2 |
| 1999 | 2 |
| 2000 | 3 |

ANS)

```

SELECT SPECIAL_YEAR.ONLYYEAR, COUNT(*) FROM (SELECT ONLYYEAR
FROM ( SELECT YEAR(JOININGDATE) ONLYYEAR , COUNT(*) ALL_DAY,
COUNT(IF(XX = 'TRUE', 1, NULL)) EVEN_MONTH_COUNT FROM
(SELECT JOININGDATE, MONTH(JOININGDATE) , CASE WHEN
MONTH(JOININGDATE)%2 = 0 THEN 'TRUE' ELSE 'FALSE' END AS XX FROM
Employee) X GROUP BY YEAR(JOININGDATE)) X WHERE
X.ALL_DAY=X.EVEN_MONTH_COUNT) SPECIAL_YEAR JOIN Employee
ON SPECIAL_YEAR.ONLYYEAR = YEAR(Employee.JOININGDATE) WHERE
DAY(Employee.JOININGDATE)%2 = 0 GROUP BY SPECIAL_YEAR.ONLYYEAR ;

```

4) Fredo and his friends regularly visit their college canteen. As with any group, on a day, one of the friends pays the canteen bill of all the friends. You are given a table of logs which shows the entries of transactions between friends. The table consists of three fields as described below:

1. P1: Name of the person who pays the bill.
2. P2: Name of the person whose bill is paid by P1.
3. amount: Amount paid by P1 for P2.

You have to summarise the transaction between all pairs of friends. See the sample input and output for explanation.

Input Format:

Table : logs

| Field | Type |
|--------|------|
| P1 | text |
| P2 | text |
| amount | int |

Output Format:

| Field | Type |
|-----------|------|
| P1 | text |
| P2 | text |
| NetAmount | int |

Sample Input: logs Table:

| P1 | P2 | amount |
|-------|-------|--------|
| Fredo | Zeus | 81 |
| Fredo | John | 59 |
| Zeus | Fredo | 81 |
| Zeus | John | 16 |
| John | Fredo | 27 |
| John | Zeus | 83 |
| Fredo | Zeus | 27 |

| | | |
|-------|------|----|
| Fredo | John | 17 |
|-------|------|----|

Sample Output:

| P1 | P2 | NetAmount |
|-------|------|-----------|
| Fredo | John | 49 |
| Fredo | Zeus | 27 |
| John | Zeus | 67 |

Explanation:

Here Fredo lends John $59+17=76$ and John lends Fredo 27. So, in all John owes Fredo 49 units. Similarly, Fredo lends Zeus $81+27=108$ and Zeus lends Fredo 81. So, in all Zeus owes Fredo 17 units. Similarly, Zeus owes John 67 units.

Note:

Only direct transactions are to be covered in the output table.

Only those entries should come in the output table which have NetAmount greater than 0. The output table should be ordered by P1 in ascending order and then by P2 in ascending order.

It is guaranteed that the input table will contain all ordered pairs of friends atleast once.

ANS)

```
select e.P1 as P1,e.P2 as P2,(e.total-f.total) as net from (select
P1,P2,sum(amount) as total from logs group by P1,P2) as e, (select
P1,P2,sum(amount) as total from logs group by P1,P2) as f where e.total-f.total
> 0 and (e.P1=f.P2 and e.P2=f.P1) order by P1 asc,P2 asc;
```