

Assignment-1 CS303

Shashank P
200010048

September 5, 2022

1 Problem 1

First we login as the root user. Then create a new username and password. We can assign privileges to the new user and login as that user.

```
1 create user 'Shashank'@'localhost' identified by 'MyPassword@123';
2 grant all privileges on lab3.* to 'Shashank'@'localhost';
3 /* Logout and login as user Shashank */
4 create database lab3;
5 use lab3;
```

```
mysql> create user 'Shashank'@'localhost' identified by 'MyPassword@123';
Query OK, 0 rows affected (0.01 sec)

mysql> grant all privileges on lab3.* to 'Shashank'@'localhost'
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds
at line 1
mysql> grant all privileges on lab3.* to 'Shashank'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> \q
Bye
shashankp@ubuntu:~/Desktop$ mysql -u Shashank -p
ERROR 1045 (28000): Access denied for user 'Shashank'@'localhost' (using password: YES)
shashankp@ubuntu:~/Desktop$ mysql -u Shashank -p'MyPassword@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.30-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Figure 1: Creation of User and adding privileges

2 Problem 2

Based on the given information, we can come up with the following schema for the tables.

Table	Primary Key	Foreign Key
part	part_no	–
supplier	supplier_no	–
shipment	shipment_no	part_no ref. part supplier_no ref. supplier

Table 1: Keys in the given Schema

```

1  create table part(
2      part_no int,
3      part_name varchar(50) not null,
4      color varchar(10),
5      weight numeric(10, 5) check(weight>0),
6      primary key (part_no)
7  );
8  create table supplier(
9      supplier_no int,
10     sup_name varchar(80) not null,
11     city varchar(25),
12     bank varchar(25),
13     primary key (supplier_no)
14 );
15 create table shipment(
16     shipment_no int,
17     part_no int,
18     supplier_no int,
19     date DATE,
20     quantity int check(quantity>=0),
21     price numeric(15, 5) check(price>=0),
22     primary key (shipment_no),
23     foreign key (part_no) references part(part_no),
24     foreign key (supplier_no) references supplier(supplier_no)
25 );

```

```

mysql> source commands.sql;
Query OK, 0 rows affected (0.04 sec)

Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+-----+
| Tables_in_lab3 |
+-----+
| part            |
| shipment        |
| supplier        |
+-----+
3 rows in set (0.00 sec)

```

Figure 2: Adding relevant tables

3 Problem 3

I have added one tuple to each table keeping in mind referential constraints.

```
1 insert into part values(15, 'Rubber', 'red', '100');
2 insert into supplier values(10, 'John', 'Paris', 'Citi-Bank');
3 insert into shipment values(20, 15, 10, '2022-09-03', 50, 10);
```

```
mysql> select * from part
-> ;
+-----+-----+-----+-----+
| part_no | part_name | color | weight |
+-----+-----+-----+-----+
|      15 | Rubber    | red   | 100.00000 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from shipment;
+-----+-----+-----+-----+-----+-----+
| shipment_no | part_no | supplier_no | date       | quantity | price |
+-----+-----+-----+-----+-----+-----+
|          20 |      15 |          10 | 2022-09-03 |        50 | 10.00000 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> select * from supplier;
+-----+-----+-----+-----+
| supplier_no | sup_name | city  | bank |
+-----+-----+-----+-----+
|          10 | John     | Paris | Citi-Bank |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Figure 3: Adding one tuple per table

4 Problem 4

Inserted multiple data points into each table. Referencial and Integrety constraints were taken care of before inserting.

```
1 insert into part values(30, 'Clip', 'yellow', '20');
2 insert into part values(45, 'Holder', 'red', '120');
3 insert into part values(60, 'Bolt', 'gray', '70');
4
5 insert into supplier values(25, 'Jane', 'Boston', 'American Bank');
6 insert into supplier values(40, 'Jack', 'New York', 'Western Union');
7
8 insert into shipment values(101, 15, 10, '2022-12-03', 18, 12.3);
9 insert into shipment values(102, 15, 25, '2020-03-04', 45, 15.6);
10 insert into shipment values(103, 15, 25, '2022-01-06', 150, 2.56);
11 insert into shipment values(104, 15, 40, '2020-05-12', 60, 24);
12
13 insert into shipment values(105, 30, 10, '2021-05-25', 100, 65);
14 insert into shipment values(106, 30, 40, '2022-04-16', 120, 12.35);
15 insert into shipment values(107, 30, 25, '2022-03-11', 50, 1.22);
16 insert into shipment values(108, 30, 10, '2021-02-28', 80, 90);
17
18 insert into shipment values(109, 45, 10, '2020-11-27', 95, 45.12);
19 insert into shipment values(110, 45, 10, '2022-10-14', 11, 120);
20 insert into shipment values(111, 45, 25, '2021-09-07', 154, 0.5);
21 insert into shipment values(112, 45, 25, '2022-09-01', 20, 1.3);
22
23 insert into shipment values(113, 60, 25, '2020-05-05', 8, 65);
24 insert into shipment values(114, 60, 40, '2021-06-04', 1, 1200);
25 insert into shipment values(115, 60, 40, '2021-08-21', 60, 100);
26 insert into shipment values(116, 60, 25, '2022-12-18', 90, 0.30);
```

```
mysql> select * from part;
```

part_no	part_name	color	weight
15	Rubber	red	100.00000
30	Clip	yellow	20.00000
45	Holder	red	120.00000
60	Bolt	gray	70.00000

```
4 rows in set (0.00 sec)
```



```
mysql> select * from supplier
-> ;
```

supplier_no	sup_name	city	bank
10	John	Paris	Citi-Bank
25	Jane	Boston	American Bank
40	Jack	New York	Western Union

```
3 rows in set (0.00 sec)
```



```
mysql> select * from shipment;
```

shipment_no	part_no	supplier_no	date	quantity	price
20	15	10	2022-09-03	50	10.00000
101	15	10	2022-12-03	18	12.30000
102	15	25	2020-03-04	45	15.60000
103	15	25	2022-01-06	150	2.56000
104	15	40	2020-05-12	60	24.00000
105	30	10	2021-05-25	100	65.00000
106	30	40	2022-04-16	120	12.35000
107	30	25	2022-03-11	50	1.22000
108	30	10	2021-02-28	80	90.00000
109	45	10	2020-11-27	95	45.12000
110	45	10	2022-10-14	11	120.00000
111	45	25	2021-09-07	154	0.50000
112	45	25	2022-09-01	20	1.30000
113	60	25	2020-05-05	8	65.00000
114	60	40	2021-06-04	1	1200.00000
115	60	40	2021-08-21	60	100.00000
116	60	25	2022-12-18	90	0.30000

```
17 rows in set (0.00 sec)
```

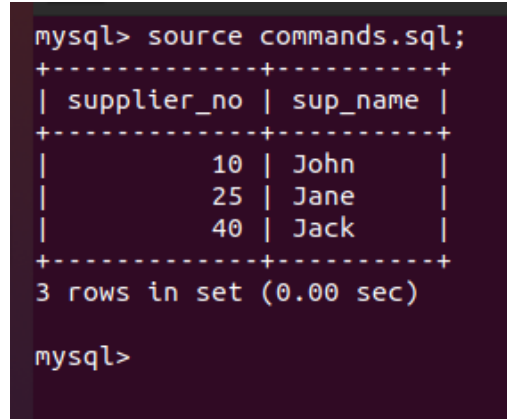
Figure 4: Adding multiple rows to each table

5 Problem 5

5.1 Part (i)

Firstly we find the natural join of all 3 tables. From this we can select names and IDs of suppliers who have sold parts with colour red.

```
1 select distinct supplier.supplier_no, supplier.sup_name
2 from (shipment natural join part) natural join supplier
3 where part.color='red';
```



```
mysql> source commands.sql;
+-----+-----+
| supplier_no | sup_name |
+-----+-----+
|          10 | John     |
|          25 | Jane     |
|          40 | Jack     |
+-----+-----+
3 rows in set (0.00 sec)

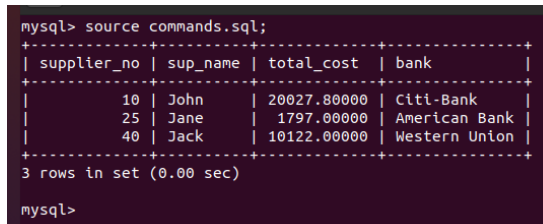
mysql>
```

Figure 5: Suppliers who have supplied red parts

5.2 Part (ii)

We can group the shipment table by *supplier_id*. We can use the aggregate *sum* function to get the total cost. Then to get the names of the suppliers, we perform a natural join with **supplier** table.

```
1 select supplier.supplier_no, supplier.sup_name, supplier_costs.total_cost, supplier.
   bank
2 from supplier natural join (
3     select shipment.supplier_no, sum(shipment.quantity * shipment.price)
4     from shipment natural join supplier
5     group by shipment.supplier_no
6 ) as supplier_costs(supplier_no, total_cost);
```



```
mysql> source commands.sql;
+-----+-----+-----+-----+
| supplier_no | sup_name | total_cost | bank      |
+-----+-----+-----+-----+
|          10 | John     | 20027.80000 | Citi-Bank |
|          25 | Jane     | 1797.00000  | American Bank |
|          40 | Jack     | 10122.00000 | Western Union |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Figure 6: Total cost of each supplier

5.3 Part (iii)

We can group the shipment table by *supplier_id*. We can use the aggregate *sum* function to get the total cost. Then to get the the names of the suppliers, we perform a natural join with **supplier** table.

```
1 with total_parts(value) as (select count(distinct part_no) from part),
2   sup_total(supplier_no, value) as (select supplier_no, count(distinct part_no) from
3   shipment group by supplier_no)
4 select supplier.supplier_no, supplier.sup_name
5 from total_parts, (supplier natural join sup_total)
where total_parts.value=sup_total.value
```

```
mysql> source commands.sql
+-----+-----+
| supplier_no | sup_name |
+-----+-----+
|          25 | Jane     |
+-----+-----+
1 row in set (0.00 sec)
```

Figure 7: Supplier who supplied all parts