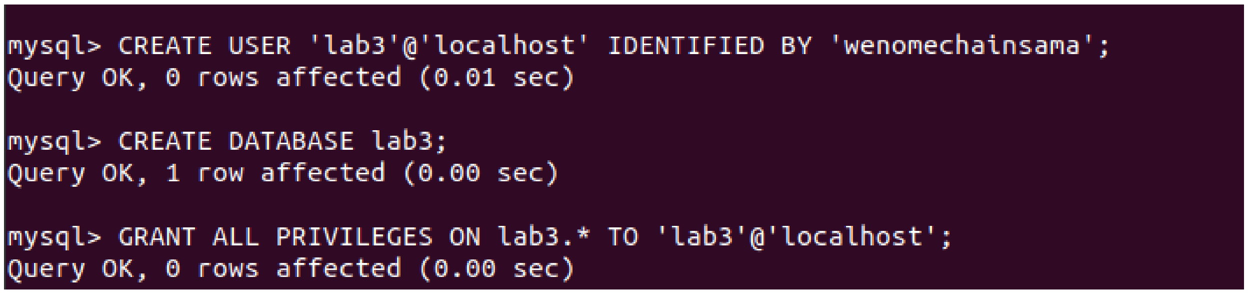
Assignment 3

Databases and Information Systems Lab

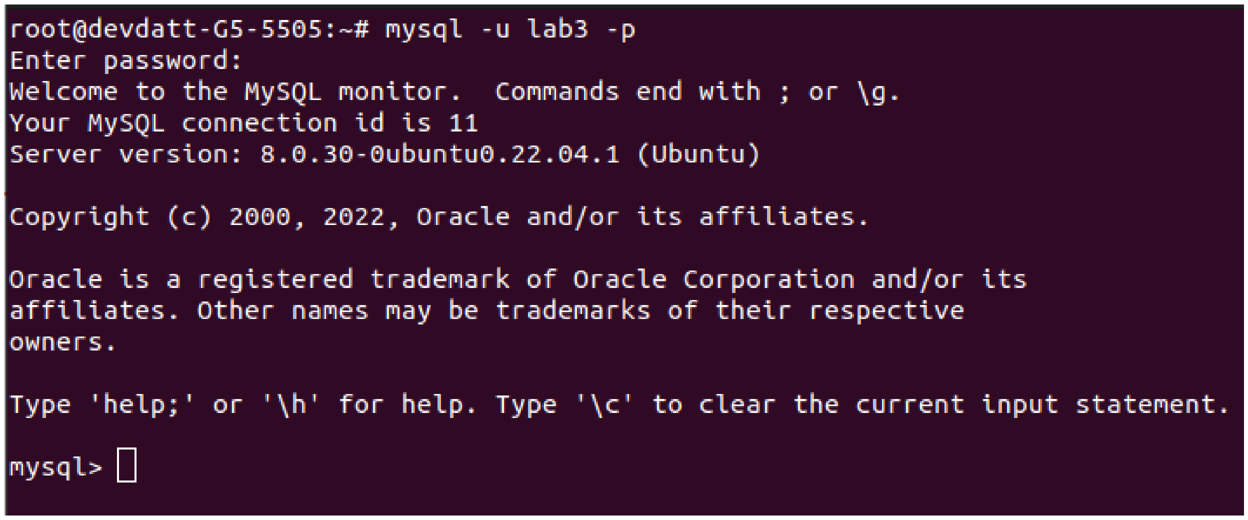
Devdatt N(200010012)

---

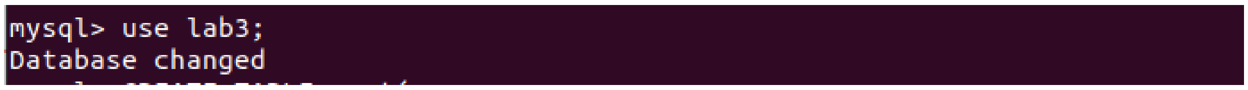
1. The queries from 200010012_1.sql are used for this. The queries and output received are as follows :



```
mysql> CREATE USER 'lab3'@'localhost' IDENTIFIED BY 'wenomechainsama';
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE DATABASE lab3;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON lab3.* TO 'lab3'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

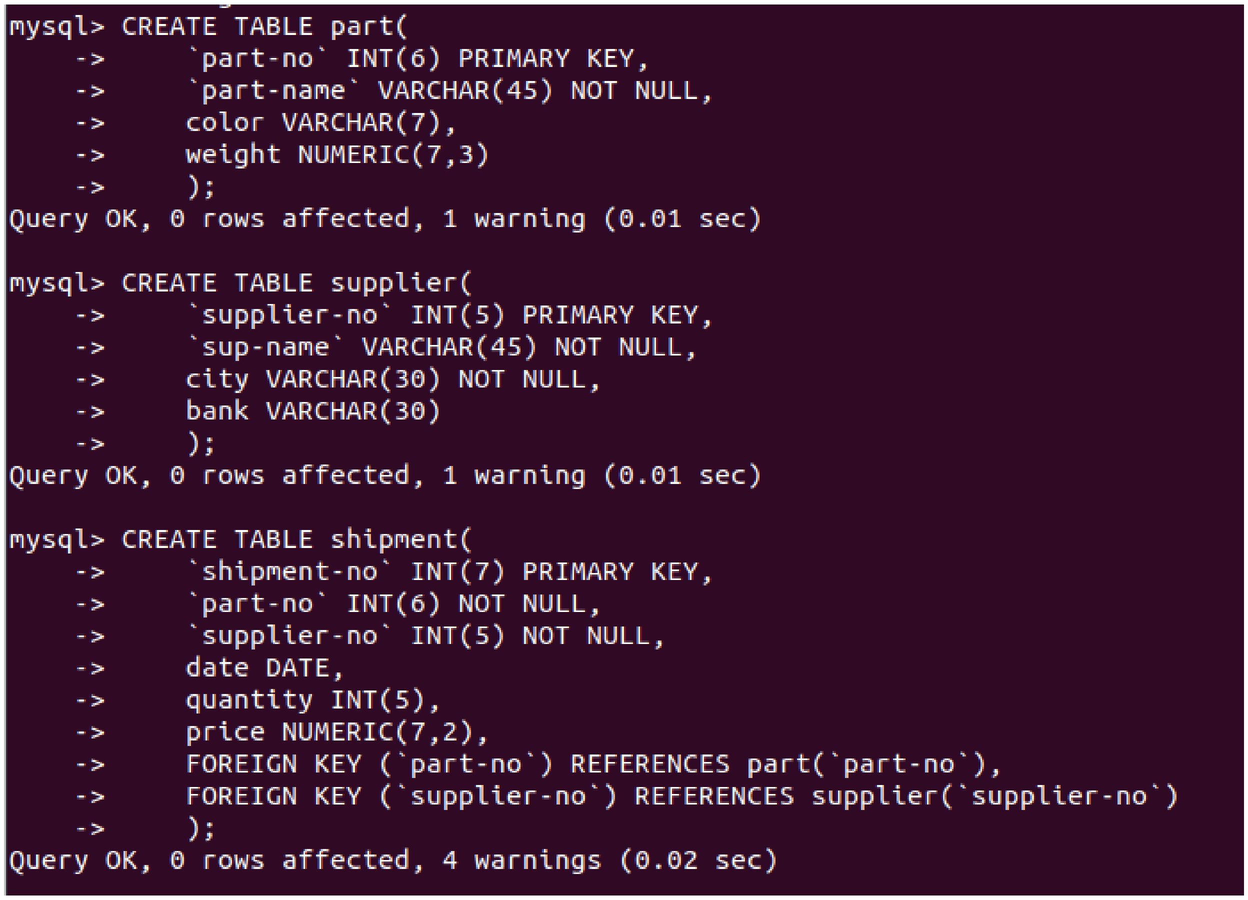


```
root@devdatt-G5-5505:~# mysql -u lab3 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.30-0ubuntu0.22.04.1 (Ubuntu)

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.

mysql> 
```



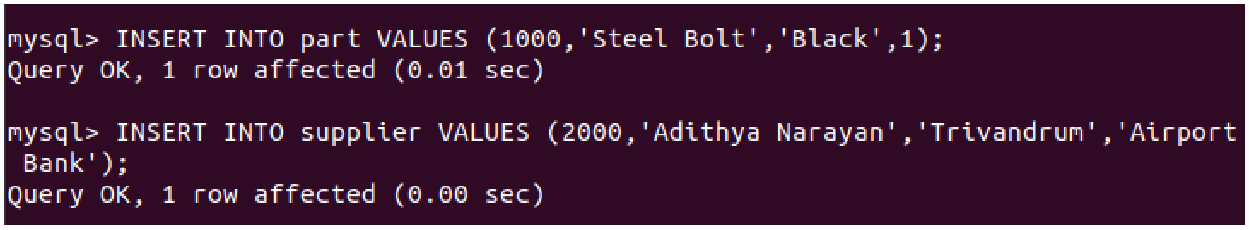
```
mysql> use lab3;
Database changed
```

2. The tables were created successfully using the queries. The queries and outputs are as follows :

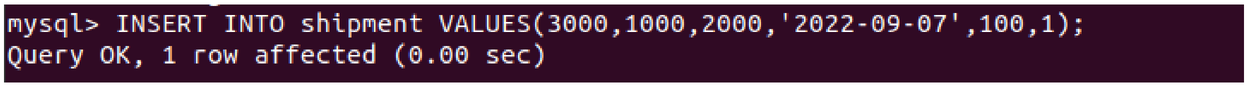


```
mysql> CREATE TABLE part(
    ->     `part-no` INT(6) PRIMARY KEY,
    ->     `part-name` VARCHAR(45) NOT NULL,
    ->     color VARCHAR(7),
    ->     weight NUMERIC(7,3)
    ->     );
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> CREATE TABLE supplier(
    ->     `supplier-no` INT(5) PRIMARY KEY,
    ->     `sup-name` VARCHAR(45) NOT NULL,
    ->     city VARCHAR(30) NOT NULL,
    ->     bank VARCHAR(30)
    ->     );
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> CREATE TABLE shipment(
    ->     `shipment-no` INT(7) PRIMARY KEY,
    ->     `part-no` INT(6) NOT NULL,
    ->     `supplier-no` INT(5) NOT NULL,
    ->     date DATE,
    ->     quantity INT(5),
    ->     price NUMERIC(7,2),
    ->     FOREIGN KEY (`part-no`) REFERENCES part(`part-no`),
    ->     FOREIGN KEY (`supplier-no`) REFERENCES supplier(`supplier-no`)
    ->     );
Query OK, 0 rows affected, 4 warnings (0.02 sec)
```

3. A tuple is added in each table as follows :

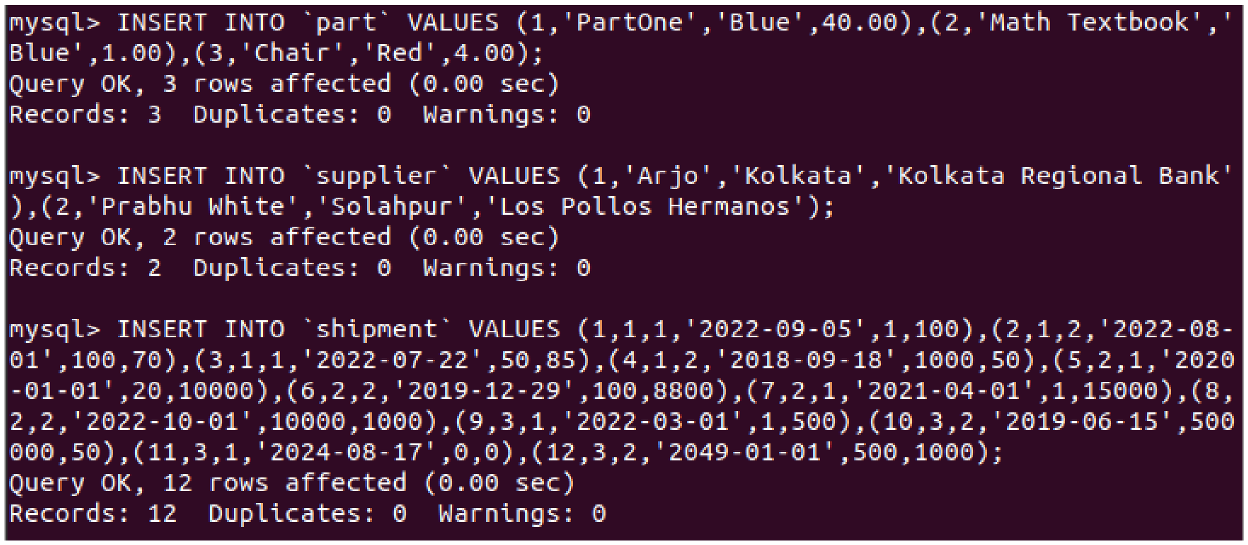


```
mysql> INSERT INTO part VALUES (1000,'Steel Bolt','Black',1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO supplier VALUES (2000,'Adithya Narayan','Trivandrum','Airport
 Bank');
Query OK, 1 row affected (0.00 sec)
```



```
mysql> INSERT INTO shipment VALUES(3000,1000,2000,'2022-09-07',100,1);
Query OK, 1 row affected (0.00 sec)
```

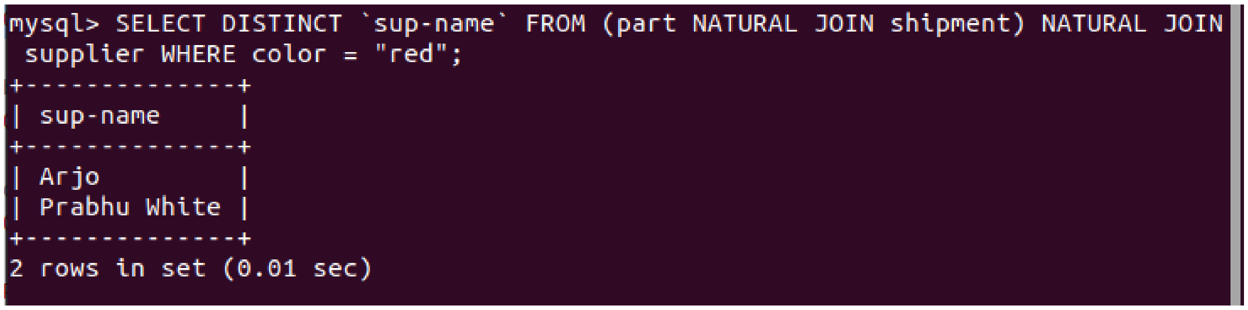
4. Values from 200010012_2.sql were inserted as follows :



```
mysql> INSERT INTO `part` VALUES (1,'PartOne','Blue',40.00),(2,'Math Textbook','
Blue',1.00),(3,'Chair','Red',4.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO `supplier` VALUES (1,'Arjo','Kolkata','Kolkata Regional Bank'
),(2,'Prabhu White','Solahpur','Los Pollos Hermanos');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO `shipment` VALUES (1,1,1,'2022-09-05',1,100),(2,1,2,'2022-08-
01',100,70),(3,1,1,'2022-07-22',50,85),(4,1,2,'2018-09-18',1000,50),(5,2,1,'2020
-01-01',20,10000),(6,2,2,'2019-12-29',100,8800),(7,2,1,'2021-04-01',1,15000),(8,
2,2,'2022-10-01',10000,1000),(9,3,1,'2022-03-01',1,500),(10,3,2,'2019-06-15',500
000,50),(11,3,1,'2024-08-17',0,0),(12,3,2,'2049-01-01',500,1000);
Query OK, 12 rows affected (0.00 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

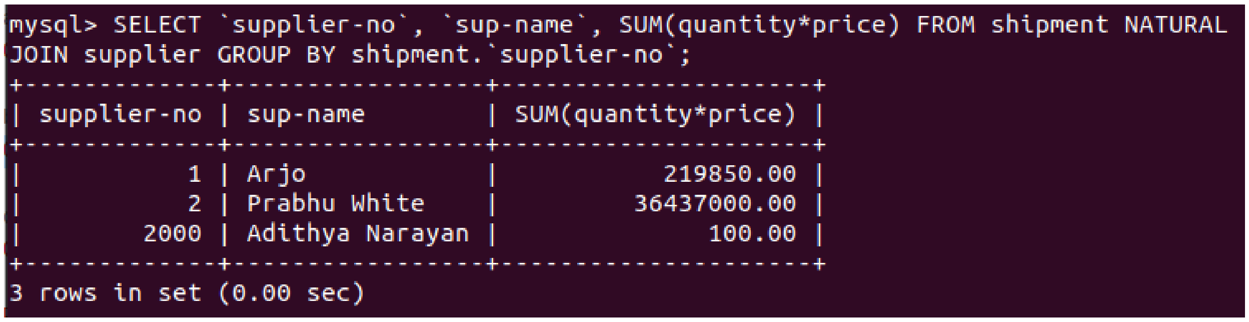
5.

a) The list of suppliers who supplied red colored parts were obtained by using natural join on the three tables and then finding distinct supplier names from the rows with red parts. This was done as follows :



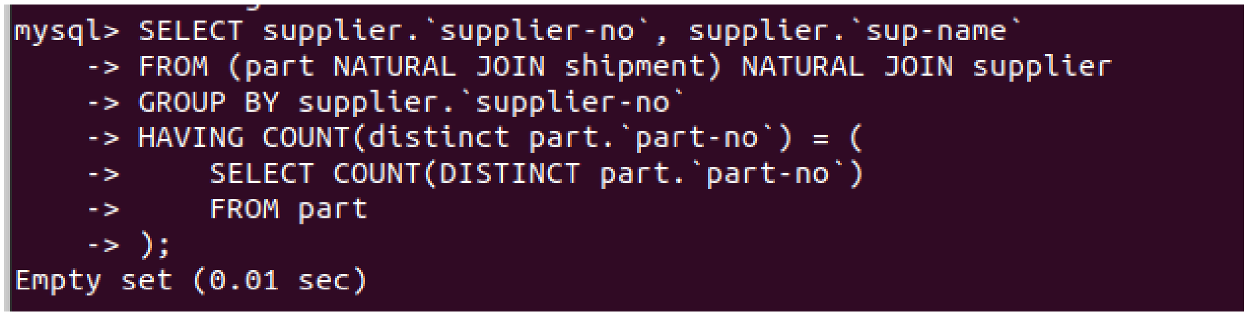
```
mysql> SELECT DISTINCT `sup-name` FROM (part NATURAL JOIN shipment) NATURAL JOIN
 supplier WHERE color = "red";
+--------------+
| sup-name     |
+--------------+
| Arjo         |
| Prabhu White |
+--------------+
2 rows in set (0.01 sec)
```

b) Here, the total price payable to each supplier was calculated with the help of natural joins and the aggregate function SUM which summed up the values obtained by multiplying ‘price’ and ‘quantity’ values. The result obtained was as follows :



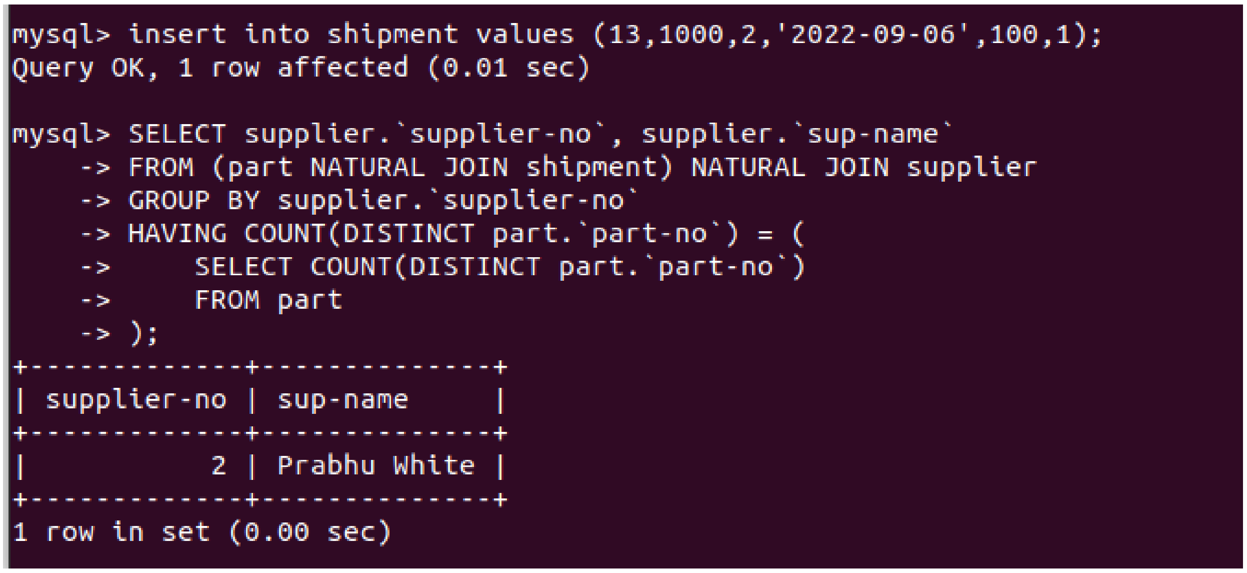
```
mysql> SELECT `supplier-no`, `sup-name`, SUM(quantity*price) FROM shipment NATURAL
JOIN supplier GROUP BY shipment.`supplier-no`;
+-------------+-----------------+---------------------+
| supplier-no | sup-name        | SUM(quantity*price) |
+-------------+-----------------+---------------------+
|           1 | Arjo            |           219850.00 |
|           2 | Prabhu White    |         36437000.00 |
|        2000 | Adithya Narayan |              100.00 |
+-------------+-----------------+---------------------+
3 rows in set (0.00 sec)
```

c) This was done by checking if the total count of distinct parts shipped by each supplier is equal to the total number of parts available. We obtained an empty set because none of our suppliers sold all parts. This was done as follows :



```
mysql> SELECT supplier.`supplier-no`, supplier.`sup-name`
    -> FROM (part NATURAL JOIN shipment) NATURAL JOIN supplier
    -> GROUP BY supplier.`supplier-no`
    -> HAVING COUNT(distinct part.`part-no`) = (
    ->     SELECT COUNT(DISTINCT part.`part-no`)
    ->     FROM part
    -> );
Empty set (0.01 sec)
```

However, to make sure, we have added one more value so that the supplier with ‘supplier-no = 2’ ends up supplying all parts. After this, we try again and results are as follows :



```
mysql> insert into shipment values (13,1000,2,'2022-09-06',100,1);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT supplier.`supplier-no`, supplier.`sup-name`
    -> FROM (part NATURAL JOIN shipment) NATURAL JOIN supplier
    -> GROUP BY supplier.`supplier-no`
    -> HAVING COUNT(DISTINCT part.`part-no`) = (
    ->     SELECT COUNT(DISTINCT part.`part-no`)
    ->     FROM part
    -> );
+-------------+--------------+
| supplier-no | sup-name     |
+-------------+--------------+
|           2 | Prabhu White |
+-------------+--------------+
1 row in set (0.00 sec)
```

Therefore, we can be sure that our queries are accurate.