Sheet No : 4

Assignment No : 08

A. Creation of Tables:

(i) Creation of Customer table:

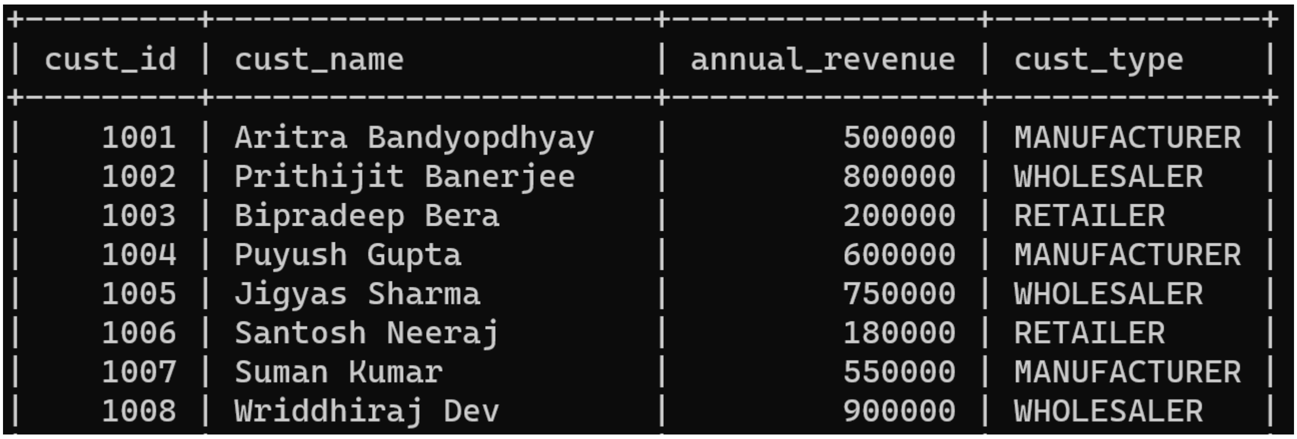
Query:

```
CREATE TABLE IF NOT EXISTS Customer (
   cust_id INT UNSIGNED,
   cust_name VARCHAR(50),
   annual_revenue INT UNSIGNED,
   cust_type ENUM('MANUFACTURER', 'WHOLESALER', 'RETAILER'),
   PRIMARY KEY (cust_id),
   CHECK (cust_id BETWEEN 100 AND 10000)
);
```

Inserting values:

```
INSERT INTO Customer (cust_id, cust_name, annual_revenue, cust_type)
VALUES
   (1001, 'Aritra Bandyopdhyay', 500000, 'MANUFACTURER'),
   (1002, 'Prithijit Banerjee', 800000, 'WHOLESALER'),
   (1003, 'Bipradeep Bera', 200000, 'RETAILER'),
   (1004, 'Puyush Gupta', 600000, 'MANUFACTURER'),
   (1005, 'Jigyas Sharma', 750000, 'WHOLESALER');
```

Snapshot of the table:



```
+---------+-----------------------+----------------+--------------+
| cust_id | cust_name             | annual_revenue | cust_type    |
+---------+-----------------------+----------------+--------------+
|    1001 | Aritra Bandyopdhyay   |         500000 | MANUFACTURER |
|    1002 | Prithijit Banerjee    |         800000 | WHOLESALER   |
|    1003 | Bipradeep Bera        |         200000 | RETAILER     |
|    1004 | Puyush Gupta          |         600000 | MANUFACTURER |
|    1005 | Jigyas Sharma         |         750000 | WHOLESALER   |
|    1006 | Santosh Neeraj        |         180000 | RETAILER     |
|    1007 | Suman Kumar           |         550000 | MANUFACTURER |
|    1008 | Wriddhiraj Dev        |         900000 | WHOLESALER   |
```

(ii) Creation of Truck table:

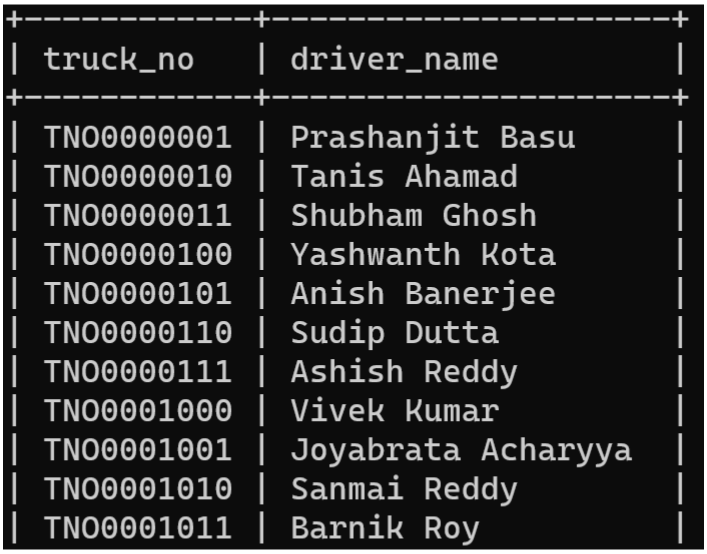
Query:

```
CREATE TABLE IF NOT EXISTS Truck (
   truck_no CHAR(10),
   driver_name VARCHAR(50),
   CHECK(truck_no LIKE 'T%'),
   PRIMARY KEY (truck_no)
);
```

Inserting values:

```
INSERT INTO Truck (truck_no, driver_name)
VALUES
 ('TNO0000001','Prashanjit Basu'),
   ('TNO0000010','Tanis Ahamad'),
   ('TNO0000011','Shubham Ghosh'),
   ('TNO0000100','Yashwanth Kota'),
   ('TNO0000101','Anish Banerjee'),
   ('TNO0000110','Sudip Dutta'),
   ('TNO0000111','Ashish Reddy');
```

Snapshot of the table:



```
+------------+---------------------+
| truck_no   | driver_name         |
+------------+---------------------+
| TNO0000001 | Prashanjit Basu     |
| TNO0000010 | Tanis Ahamad        |
| TNO0000011 | Shubham Ghosh       |
| TNO0000100 | Yashwanth Kota      |
| TNO0000101 | Anish Banerjee      |
| TNO0000110 | Sudip Dutta         |
| TNO0000111 | Ashish Reddy        |
| TNO0001000 | Vivek Kumar         |
| TNO0001001 | Joyabrata Acharyya  |
| TNO0001010 | Sanmai Reddy        |
| TNO0001011 | Barnik Roy          |
```

(iii) Creation of City table:

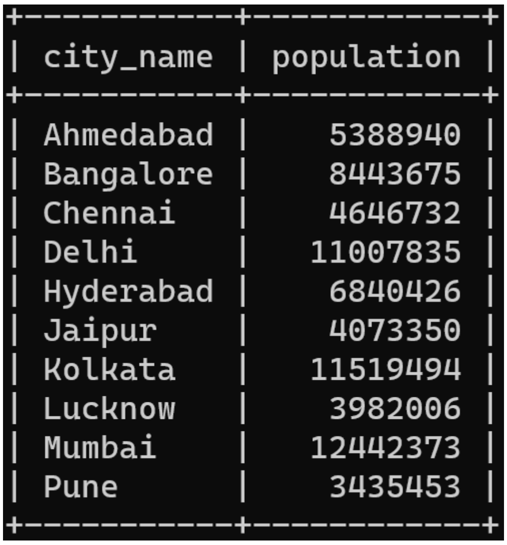
Query:

```
CREATE TABLE IF NOT EXISTS City (
   city_name VARCHAR(50),
   population INT UNSIGNED,
   PRIMARY KEY (city_name)
);
```

Inserting values:

```
INSERT INTO City (city_name, population)
VALUES
   ('Mumbai', 12442373),
   ('Delhi', 11007835),
   ('Bangalore', 8443675),
   ('Kolkata', 4949498),
   ('Chennai', 4646732),
   ('Hyderabad', 6759413),
   ('Pune', 3124453),
   ('Ahmedabad', 5577940),
   ('Jaipur', 3073350),
   ('Lucknow', 3382000);
```

Snapshot of the table:



```
+-----------+------------+
| city_name | population |
+-----------+------------+
| Ahmedabad |    5388940 |
| Bangalore |    8443675 |
| Chennai   |    4646732 |
| Delhi     |   11007835 |
| Hyderabad |    6840426 |
| Jaipur    |    4073350 |
| Kolkata   |   11519494 |
| Lucknow   |    3982006 |
| Mumbai    |   12442373 |
| Pune      |    3435453 |
+-----------+------------+
```

(iv) Creation of Shipment table:

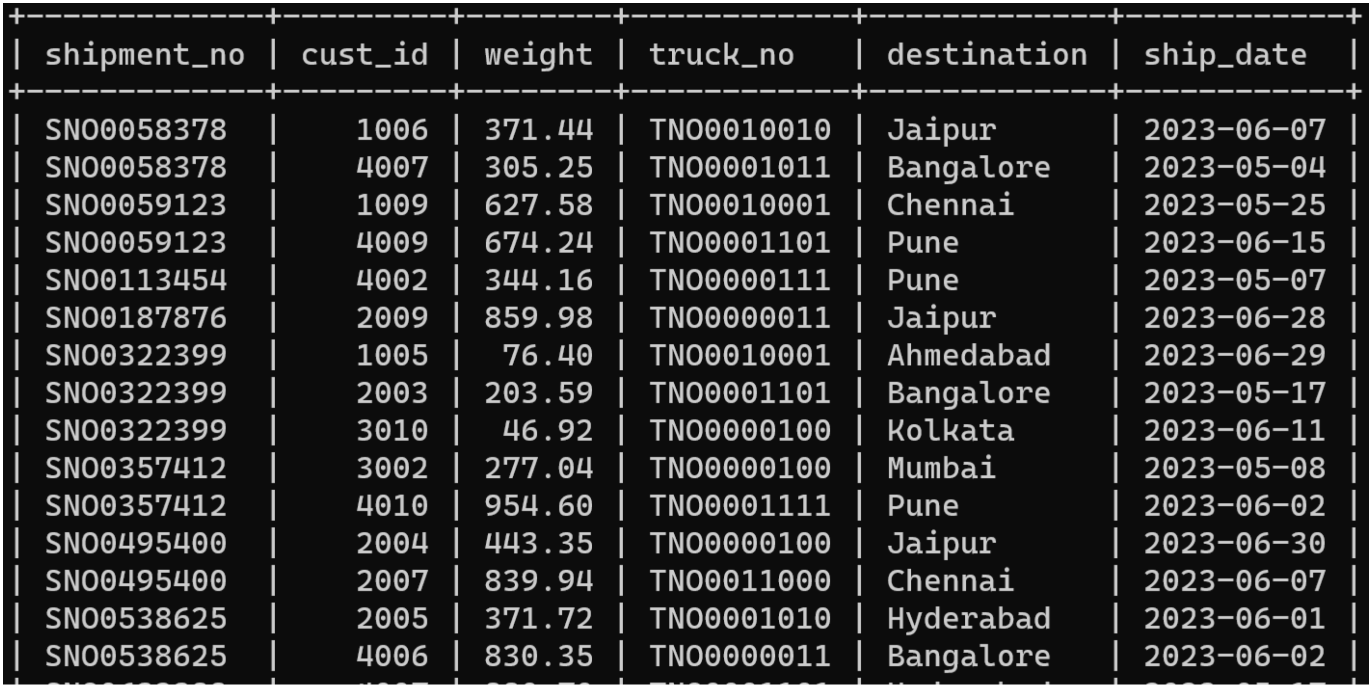
Query:

```
CREATE TABLE IF NOT EXISTS Shipment (
   shipment_no CHAR(10),
   cust_id INT UNSIGNED,
    weight DECIMAL(5, 2),
    truck_no CHAR(10),
    destination VARCHAR(50),
    ship_date DATE,
    CHECK (weight >= 0 AND weight < 1000),
    CHECK (shipment_no LIKE 'S%'),
    PRIMARY KEY (shipment_no, cust_id),
    FOREIGN KEY (cust_id) REFERENCES Customer(cust_id) ON DELETE CASCADE,
    FOREIGN KEY (truck_no) REFERENCES Truck(truck_no) ON DELETE SET NULL,
    FOREIGN KEY (destination) REFERENCES City(city_name)
);
```

Inserting values:

```
INSERT INTO Shipment (shipment_no, cust_id, weight, truck_no, destination,
ship_date)
VALUES
        ('SNO2709268', 4004, 230.39, 'TNO0010000', 'Jaipur', '2023-06-02'),
        ('SNO2709268', 3003, 665.34, 'TNO0001010', 'Hyderabad', '2023-05-15'),
        ('SNO2709268', 2002, 319.97, 'TNO0010110', 'Jaipur', '2023-05-09'),
        ('SNO4761770', 3001, 969.62, 'TNO0001010', 'Pune', '2023-06-20'),
        ('SNO4761770', 3004, 947.81, 'TNO0011000', 'Ahmedabad', '2023-05-
        26'),
        ('SNO0187876', 2009, 859.98, 'TNO0000011', 'Jaipur', '2023-06-28'),
        ('SNO4156545', 2001, 703.03, 'TNO0010111', 'Chennai', '2023-06-27'),
        ('SNO5060006', 2006, 777.32, 'TNO0000101', 'Kolkata', '2023-05-10'),
        ('SNO5060006', 1003, 250.69, 'TNO0011000', 'Jaipur', '2023-06-28');
```

Snapshot of the table:



```
+-------------+---------+--------+------------+-------------+------------+
| shipment_no | cust_id | weight | truck_no   | destination | ship_date  |
+-------------+---------+--------+------------+-------------+------------+
| SNO0058378  |    1006 | 371.44 | TNO0010010 | Jaipur      | 2023-06-07 |
| SNO0058378  |    4007 | 305.25 | TNO0001011 | Bangalore   | 2023-05-04 |
| SNO0059123  |    1009 | 627.58 | TNO0010001 | Chennai     | 2023-05-25 |
| SNO0059123  |    4009 | 674.24 | TNO0001101 | Pune        | 2023-06-15 |
| SNO0113454  |    4002 | 344.16 | TNO0000111 | Pune        | 2023-05-07 |
| SNO0187876  |    2009 | 859.98 | TNO0000011 | Jaipur      | 2023-06-28 |
| SNO0322399  |    1005 |  76.40 | TNO0010001 | Ahmedabad   | 2023-06-29 |
| SNO0322399  |    2003 | 203.59 | TNO0001101 | Bangalore   | 2023-05-17 |
| SNO0322399  |    3010 |  46.92 | TNO0000100 | Kolkata     | 2023-06-11 |
| SNO0357412  |    3002 | 277.04 | TNO0000100 | Mumbai      | 2023-05-08 |
| SNO0357412  |    4010 | 954.60 | TNO0001111 | Pune        | 2023-06-02 |
| SNO0495400  |    2004 | 443.35 | TNO0000100 | Jaipur      | 2023-06-30 |
| SNO0495400  |    2007 | 839.94 | TNO0011000 | Chennai     | 2023-06-07 |
| SNO0538625  |    2005 | 371.72 | TNO0001010 | Hyderabad   | 2023-06-01 |
| SNO0538625  |    4006 | 830.35 | TNO0000011 | Bangalore   | 2023-06-02 |
```

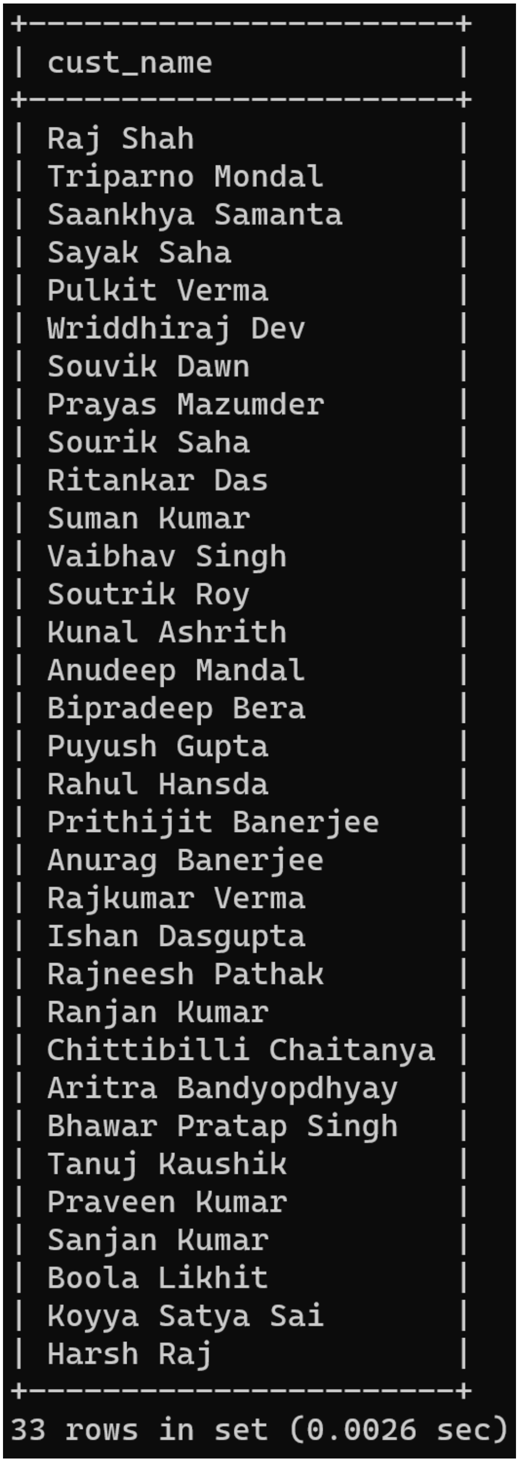
B. Queries and their solutions:

1) Give names of customer who have sent packages (shipments) to Kolkata, Chennai and Mumbai.

Query:

```
select distinct cust_name
from shipment s, customer c
where s.cust_id = c.cust_id
and destination in (‘Kolkata’, ‘Chennai’, ‘Mumbai’);
```

Output:



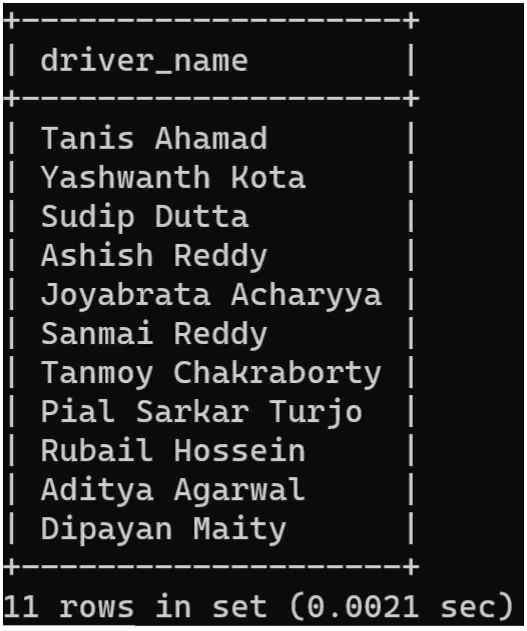
```
+-----------------------+
| cust_name             |
+-----------------------+
| Raj Shah              |
| Triparno Mondal       |
| Saankhya Samanta      |
| Sayak Saha            |
| Pulkit Verma          |
| Wriddhiraj Dev        |
| Souvik Dawn           |
| Prayas Mazumder       |
| Sourik Saha           |
| Ritankar Das          |
| Suman Kumar           |
| Vaibhav Singh         |
| Soutrik Roy           |
| Kunal Ashrith         |
| Anudeep Mandal        |
| Bipradeep Bera        |
| Puyush Gupta          |
| Rahul Hansda          |
| Prithijit Banerjee    |
| Anurag Banerjee       |
| Rajkumar Verma        |
| Ishan Dasgupta        |
| Rajneesh Pathak       |
| Ranjan Kumar          |
| Chittibilli Chaitanya |
| Aritra Bandyopdhyay   |
| Bhawar Pratap Singh   |
| Tanuj Kaushik         |
| Praveen Kumar         |
| Sanjan Kumar          |
| Boola Likhit          |
| Koyya Satya Sai       |
| Harsh Raj             |
+-----------------------+
33 rows in set (0.0026 sec)
```

2) **List the names of the driver who have delivered shipments weighing over 900 pounds.**

Query:

```
select distinct driver_name
from shipment s, truck t
where s.truck_no = t.truck_no
and s.weight > 900;
```

Output:



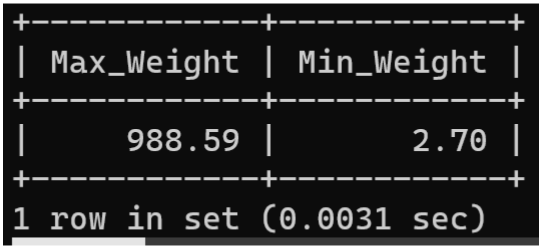
```
+--------------------+
| driver_name        |
+--------------------+
| Tanis Ahamad       |
| Yashwanth Kota     |
| Sudip Dutta        |
| Ashish Reddy       |
| Joyabrata Acharyya |
| Sanmai Reddy       |
| Tanmoy Chakraborty |
| Pial Sarkar Turjo  |
| Rubail Hossein     |
| Aditya Agarwal     |
| Dipayan Maity      |
+--------------------+
11 rows in set (0.0021 sec)
```

3) **Retrieve the maximum and minimum weights of the shipments. Rename the output as Max_Weight and Min_Weight respectively.**

Query:

```
select
    max(weight) as Max_Weight,
    min(weight) as Min_Weight
from shipment;
```

Output:



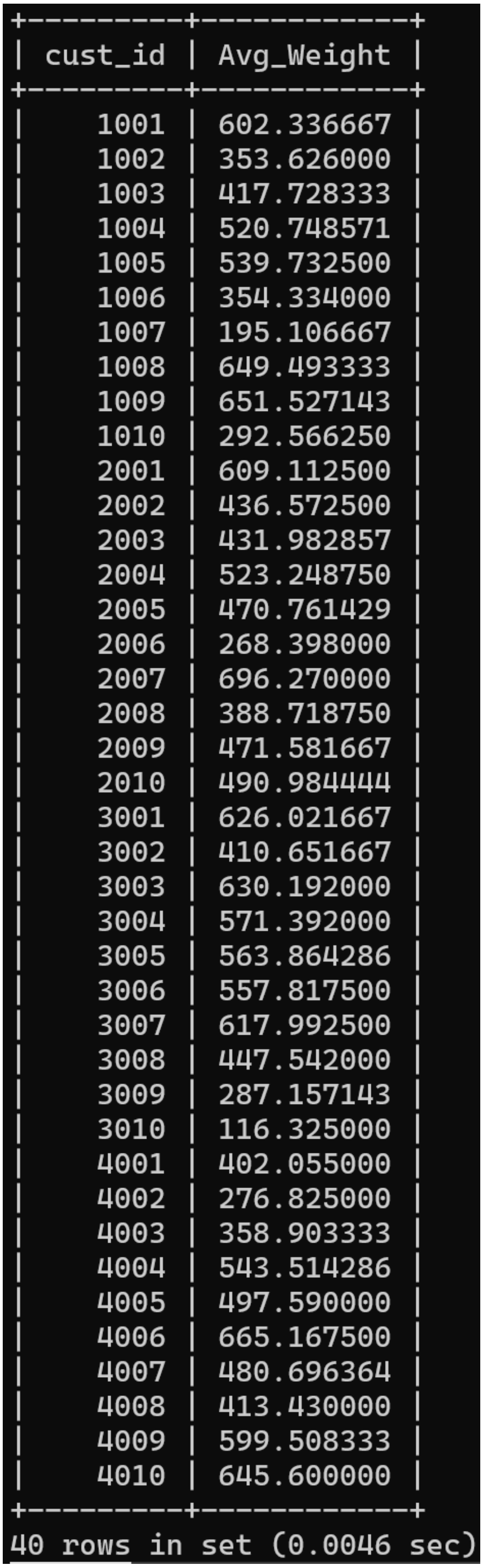
```
+------------+------------+
| Max_Weight | Min_Weight |
+------------+------------+
|     988.59 |       2.70 |
+------------+------------+
1 row in set (0.0031 sec)
```

4) **For each customer, what is the average weight of package sent by the customer?**

Query:

```
select cust_id, avg(weight) as Avg_Weight
from shipment
group by cust_id
Order by cust_id;
```

<u>Output:</u>



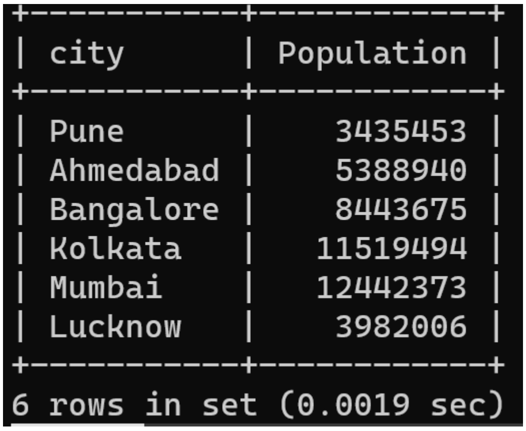
```
+---------+------------+
| cust_id | Avg_Weight |
+---------+------------+
|    1001 | 602.336667 |
|    1002 | 353.626000 |
|    1003 | 417.728333 |
|    1004 | 520.748571 |
|    1005 | 539.732500 |
|    1006 | 354.334000 |
|    1007 | 195.106667 |
|    1008 | 649.493333 |
|    1009 | 651.527143 |
|    1010 | 292.566250 |
|    2001 | 609.112500 |
|    2002 | 436.572500 |
|    2003 | 431.982857 |
|    2004 | 523.248750 |
|    2005 | 470.761429 |
|    2006 | 268.398000 |
|    2007 | 696.270000 |
|    2008 | 388.718750 |
|    2009 | 471.581667 |
|    2010 | 490.984444 |
|    3001 | 626.021667 |
|    3002 | 410.651667 |
|    3003 | 630.192000 |
|    3004 | 571.392000 |
|    3005 | 563.864286 |
|    3006 | 557.817500 |
|    3007 | 617.992500 |
|    3008 | 447.542000 |
|    3009 | 287.157143 |
|    3010 | 116.325000 |
|    4001 | 402.055000 |
|    4002 | 276.825000 |
|    4003 | 358.903333 |
|    4004 | 543.514286 |
|    4005 | 497.590000 |
|    4006 | 665.167500 |
|    4007 | 480.696364 |
|    4008 | 413.430000 |
|    4009 | 599.508333 |
|    4010 | 645.600000 |
+---------+------------+
40 rows in set (0.0046 sec)
```

5) List the names and populations of cities that have received a shipment weighing over 900 pounds.

Query:

```
select distinct s.destination as city, c.population as population
from shipment s, (select city_name, population from city) c
where s.destination = c.city_name
and s.weight > 900
group by s.destination;
```

Output:



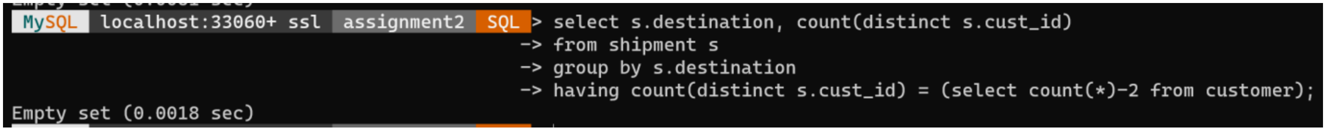
```
+-----------+------------+
| city      | Population |
+-----------+------------+
| Pune      |    3435453 |
| Ahmedabad |    5388940 |
| Bangalore |    8443675 |
| Kolkata   |   11519494 |
| Mumbai    |   12442373 |
| Lucknow   |    3982006 |
+-----------+------------+
6 rows in set (0.0019 sec)
```

6) List cities that have received shipments from every customer.

Query:

```
select s.destination, count(distinct s.cust_id)
from shipment s
group by s.destination
having count(distinct s.cust_id) = (select count(*)-2 from customer);
```

Output:



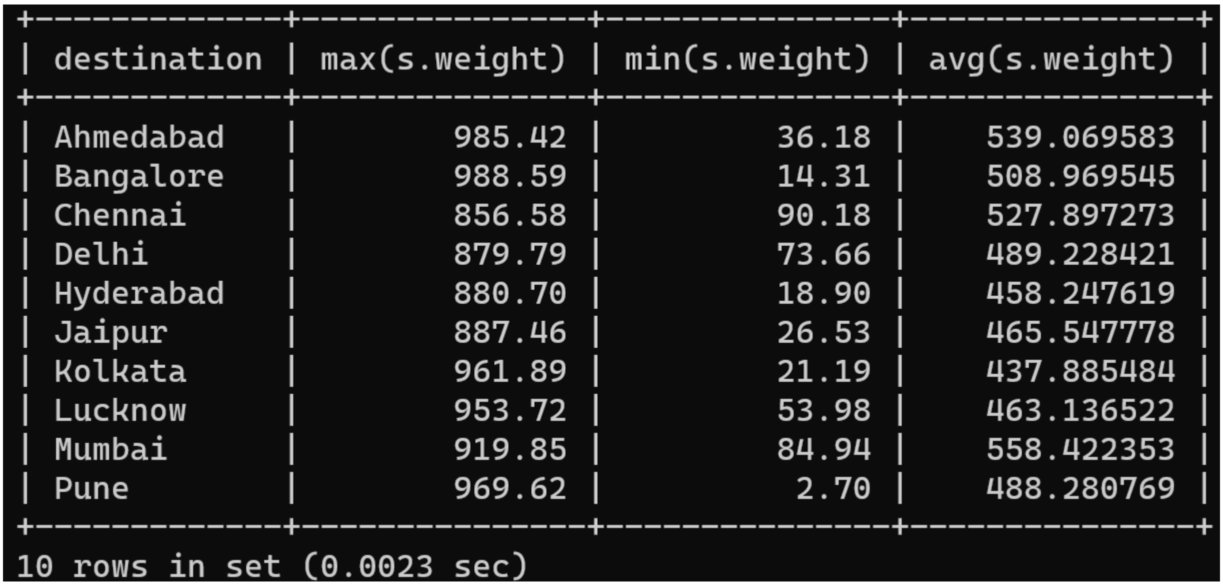
```
MySQL  localhost:33060+ ssl  assignment2  SQL > select s.destination, count(distinct s.cust_id)
                                            -> from shipment s
                                            -> group by s.destination
                                            -> having count(distinct s.cust_id) = (select count(*)-2 from customer);
Empty set (0.0018 sec)
```

7) For each city, what is the maximum weight of a package sent to that city?

Query:

```
select s.destination, max(s.weight), min(s.weight), avg(s.weight)
from shipment s
group by s.destination;
```

Output:



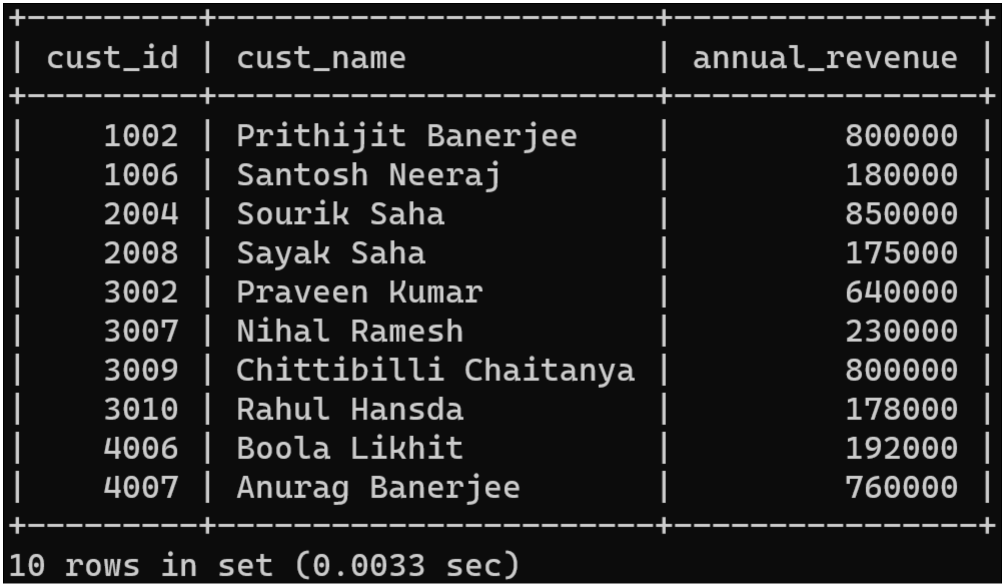
```
+-------------+---------------+---------------+---------------+
| destination | max(s.weight) | min(s.weight) | avg(s.weight) |
+-------------+---------------+---------------+---------------+
| Ahmedabad   |        985.42 |         36.18 |    539.069583 |
| Bangalore   |        988.59 |         14.31 |    508.969545 |
| Chennai     |        856.58 |         90.18 |    527.897273 |
| Delhi       |        879.79 |         73.66 |    489.228421 |
| Hyderabad   |        880.70 |         18.90 |    458.247619 |
| Jaipur      |        887.46 |         26.53 |    465.547778 |
| Kolkata     |        961.89 |         21.19 |    437.885484 |
| Lucknow     |        953.72 |         53.98 |    463.136522 |
| Mumbai      |        919.85 |         84.94 |    558.422353 |
| Pune        |        969.62 |          2.70 |    488.280769 |
+-------------+---------------+---------------+---------------+
10 rows in set (0.0023 sec)
```

8) **<u>List the name and annual revenue of customers whose shipments have been delivered by truck driver ‘Yashwanth Kota’.</u>**

<u>Query:</u>

```
select distinct c.cust_id, c.cust_name, c.annual_revenue
from customer c
where c.cust_id in (
   select distinct s.cust_id
   from shipment s
   where s.truck_no in (
      select t.truck_no
      from truck t
      where t.driver_name = 'yashwanth kota'
   )
)
order by c.cust_id;
```

<u>Output:</u>



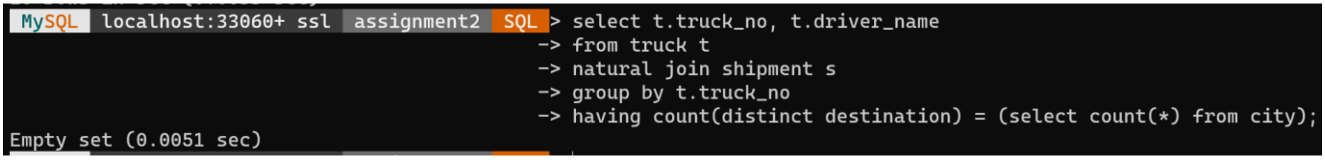
```
+---------+-----------------------+----------------+
| cust_id | cust_name             | annual_revenue |
+---------+-----------------------+----------------+
|    1002 | Prithijit Banerjee    |         800000 |
|    1006 | Santosh Neeraj        |         180000 |
|    2004 | Sourik Saha           |         850000 |
|    2008 | Sayak Saha            |         175000 |
|    3002 | Praveen Kumar         |         640000 |
|    3007 | Nihal Ramesh          |         230000 |
|    3009 | Chittibilli Chaitanya |         800000 |
|    3010 | Rahul Hansda          |         178000 |
|    4006 | Boola Likhit          |         192000 |
|    4007 | Anurag Banerjee       |         760000 |
+---------+-----------------------+----------------+
10 rows in set (0.0033 sec)
```

9) List drivers who have delivered shipments to every city.

Query:

```
select t.truck_no, t.driver_name
from truck t
where exists (
   select 1
   from shipment s
   where s.truck_no = t.truck_no
   group by s.truck_no
   having count(distinct s.destination) = (select count(*) from city)
);
```

Output:



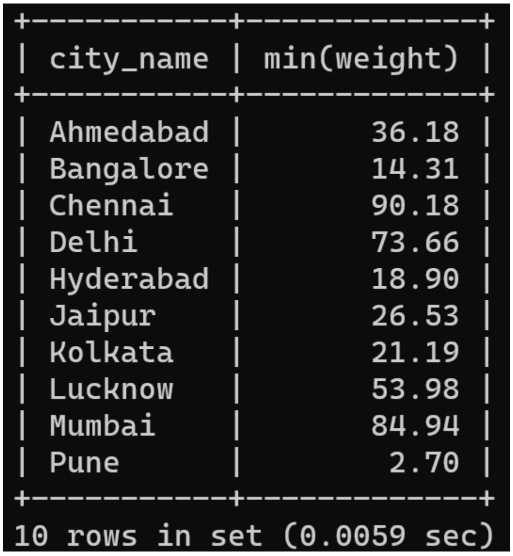
```
MySQL  localhost:33060+ ssl  assignment2  SQL > select t.truck_no, t.driver_name
                                             -> from truck t
                                             -> natural join shipment s
                                             -> group by t.truck_no
                                             -> having count(distinct destination) = (select count(*) from city);
Empty set (0.0051 sec)
```

10) For each city, with population over 1 million, what is the minimum weight of a package sent to that city.

Query:

```
select
   city_name, min(weight)
from
   Shipment, City
where
   destination = city_name
   and
   population > 1000000;
group by city_name;
```

Output:



```
+-----------+-------------+
| city_name | min(weight) |
+-----------+-------------+
| Ahmedabad |       36.18 |
| Bangalore |       14.31 |
| Chennai   |       90.18 |
| Delhi     |       73.66 |
| Hyderabad |       18.90 |
| Jaipur    |       26.53 |
| Kolkata   |       21.19 |
| Lucknow   |       53.98 |
| Mumbai    |       84.94 |
| Pune      |        2.70 |
+-----------+-------------+
10 rows in set (0.0059 sec)
```

Assignment No : 09

A. Creation of Tables:

(i) Creation of EMP Table:

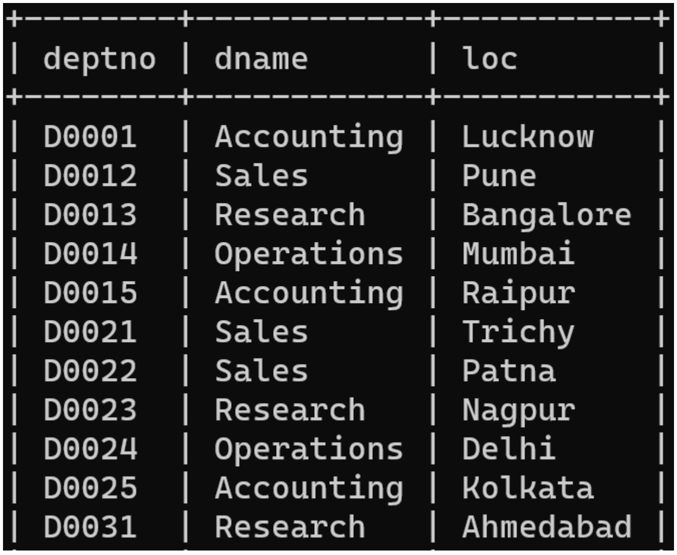
Query:

```
CREATE TABLE IF NOT EXISTS dept (
   deptno CHAR(5) PRIMARY KEY CHECK (deptno LIKE 'D%'),
   dname ENUM('Accounting', 'Sales', 'Research', 'Operations'),
   loc VARCHAR(20)
);
```

Inserting values:

```
INSERT INTO dept (deptno, dname, loc) VALUES
    ('D0001', 'Accounting', 'Lucknow'),
    ('D0012', 'Sales', 'Pune'),
    ('D0013', 'Research', 'Bangalore'),
    ('D0014', 'Operations', 'Mumbai'),
    ('D0015', 'Accounting', 'Raipur'),
    ('D0021', 'Sales', 'Trichy'),
    ('D0023', 'Research', 'Nagpur');
```

Snapshot of the table:



```
+--------+------------+-----------+
| deptno | dname      | loc       |
+--------+------------+-----------+
| D0001  | Accounting | Lucknow   |
| D0012  | Sales      | Pune      |
| D0013  | Research   | Bangalore |
| D0014  | Operations | Mumbai    |
| D0015  | Accounting | Raipur    |
| D0021  | Sales      | Trichy    |
| D0022  | Sales      | Patna     |
| D0023  | Research   | Nagpur    |
| D0024  | Operations | Delhi     |
| D0025  | Accounting | Kolkata   |
| D0031  | Research   | Ahmedabad |
```

(ii) Creation of DEPT Table:

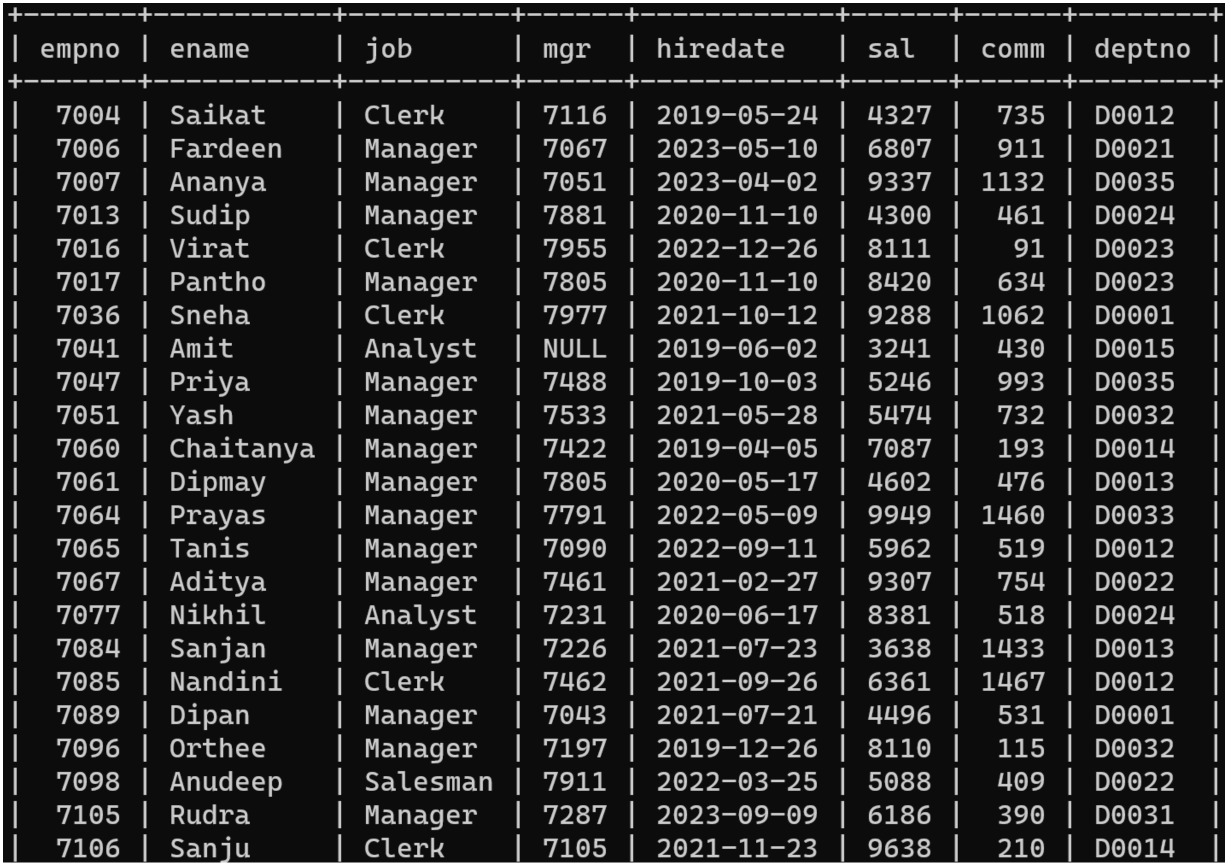
Query:

```
CREATE TABLE IF NOT EXISTS emp (
    empno INT PRIMARY KEY CHECK (empno BETWEEN 7000 AND 8000),
    ename VARCHAR(10),
    job ENUM('Clerk', 'Salesman', 'Manager', 'Analyst', 'President'),
    mgr INT REFERENCES emp(empno),
    hiredate DATE,
    sal DECIMAL(10, 0),
    comm DECIMAL(4, 0) DEFAULT 0 CHECK (comm < 1500),
    deptno CHAR(5) REFERENCES dept(deptno)
);
```

Inserting values:

```
INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES
    (7444, 'Anish', 'Manager', 7439, '2019-07-06', 4327, 1333, 'D0034'),
    (7890, 'Subhra', 'Manager', 7157, '2021-11-01', 9026, 1378, 'D0025'),
    (7784, 'Achuth', 'Manager', 7585, '2022-02-19', 3454, 469, 'D0022'),
    (7182, 'Devika', 'Manager', 7137, '2020-04-23', 9091, 49, 'D0031'),
    (7681, 'Rishabh', 'Manager', 7046, '2023-05-04', 4388, 31, 'D0031'),
    (7013, 'Sudip', 'Manager', 7881, '2020-11-10', 4300, 461, 'D0024'),
    (7617, 'Ashish', 'Manager', 7182, '2021-04-13', 6923, 387, 'D0015');
```

Snapshot of the table:



```
+-------+-----------+----------+------+------------+------+------+--------+
| empno | ename     | job      | mgr  | hiredate   | sal  | comm | deptno |
+-------+-----------+----------+------+------------+------+------+--------+
|  7004 | Saikat    | Clerk    | 7116 | 2019-05-24 | 4327 |  735 | D0012  |
|  7006 | Fardeen   | Manager  | 7067 | 2023-05-10 | 6807 |  911 | D0021  |
|  7007 | Ananya    | Manager  | 7051 | 2023-04-02 | 9337 | 1132 | D0035  |
|  7013 | Sudip     | Manager  | 7881 | 2020-11-10 | 4300 |  461 | D0024  |
|  7016 | Virat     | Clerk    | 7955 | 2022-12-26 | 8111 |   91 | D0023  |
|  7017 | Pantho    | Manager  | 7805 | 2020-11-10 | 8420 |  634 | D0023  |
|  7036 | Sneha     | Clerk    | 7977 | 2021-10-12 | 9288 | 1062 | D0001  |
|  7041 | Amit      | Analyst  | NULL | 2019-06-02 | 3241 |  430 | D0015  |
|  7047 | Priya     | Manager  | 7488 | 2019-10-03 | 5246 |  993 | D0035  |
|  7051 | Yash      | Manager  | 7533 | 2021-05-28 | 5474 |  732 | D0032  |
|  7060 | Chaitanya | Manager  | 7422 | 2019-04-05 | 7087 |  193 | D0014  |
|  7061 | Dipmay    | Manager  | 7805 | 2020-05-17 | 4602 |  476 | D0013  |
|  7064 | Prayas    | Manager  | 7791 | 2022-05-09 | 9949 | 1460 | D0033  |
|  7065 | Tanis     | Manager  | 7090 | 2022-09-11 | 5962 |  519 | D0012  |
|  7067 | Aditya    | Manager  | 7461 | 2021-02-27 | 9307 |  754 | D0022  |
|  7077 | Nikhil    | Analyst  | 7231 | 2020-06-17 | 8381 |  518 | D0024  |
|  7084 | Sanjan    | Manager  | 7226 | 2021-07-23 | 3638 | 1433 | D0013  |
|  7085 | Nandini   | Clerk    | 7462 | 2021-09-26 | 6361 | 1467 | D0012  |
|  7089 | Dipan     | Manager  | 7043 | 2021-07-21 | 4496 |  531 | D0001  |
|  7096 | Orthee    | Manager  | 7197 | 2019-12-26 | 8110 |  115 | D0032  |
|  7098 | Anudeep   | Salesman | 7911 | 2022-03-25 | 5088 |  409 | D0022  |
|  7105 | Rudra     | Manager  | 7287 | 2023-09-09 | 6186 |  390 | D0031  |
|  7106 | Sanju     | Clerk    | 7105 | 2021-11-23 | 9638 |  210 | D0014  |
```

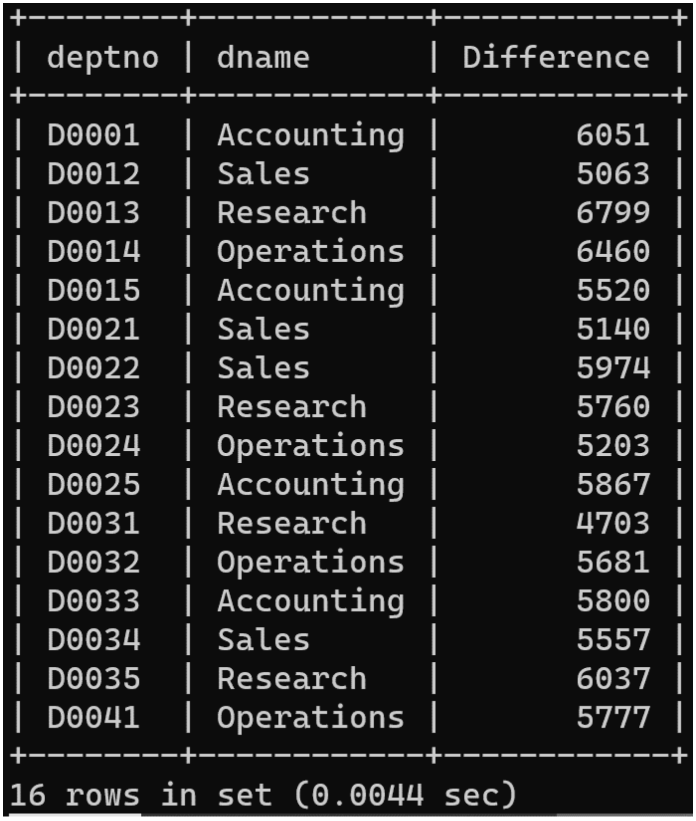
B. Queries and their solutions:

1. **Display the difference between highest and lowest salary of each department in descending order. Label the column as "Difference".**

Query:

```
select d.deptno, d.dname,
     (select max(sal) from emp where emp.deptno = d.deptno) -
     (select min(sal) from emp where emp.deptno = d.deptno) as difference
from dept d;
```

Output:



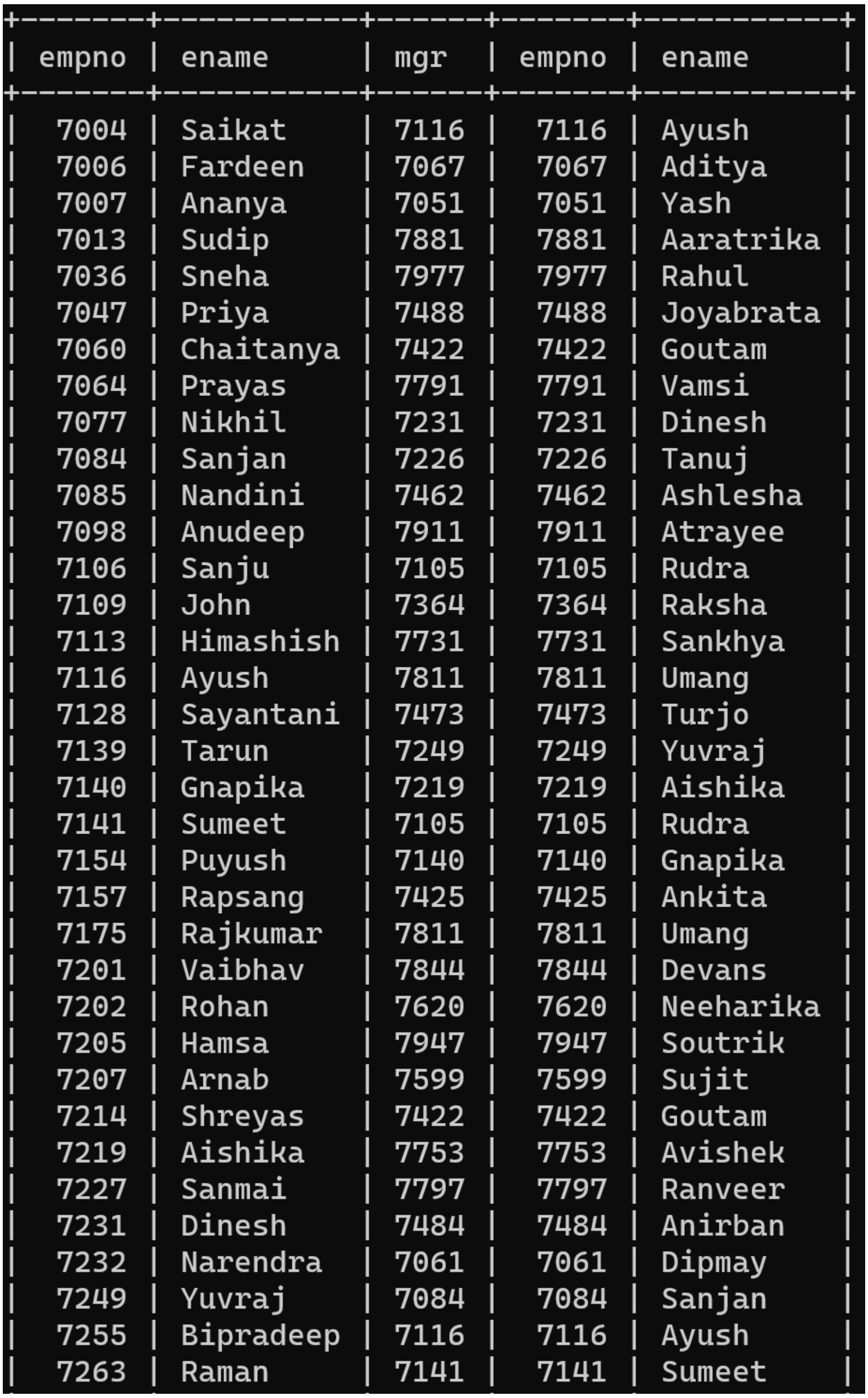
```
+--------+------------+------------+
| deptno | dname      | Difference |
+--------+------------+------------+
| D0001  | Accounting |       6051 |
| D0012  | Sales      |       5063 |
| D0013  | Research   |       6799 |
| D0014  | Operations |       6460 |
| D0015  | Accounting |       5520 |
| D0021  | Sales      |       5140 |
| D0022  | Sales      |       5974 |
| D0023  | Research   |       5760 |
| D0024  | Operations |       5203 |
| D0025  | Accounting |       5867 |
| D0031  | Research   |       4703 |
| D0032  | Operations |       5681 |
| D0033  | Accounting |       5800 |
| D0034  | Sales      |       5557 |
| D0035  | Research   |       6037 |
| D0041  | Operations |       5777 |
+--------+------------+------------+
16 rows in set (0.0044 sec)
```

2. **List all the employees' employee number and name along with their immediate managers' employee number and name.**

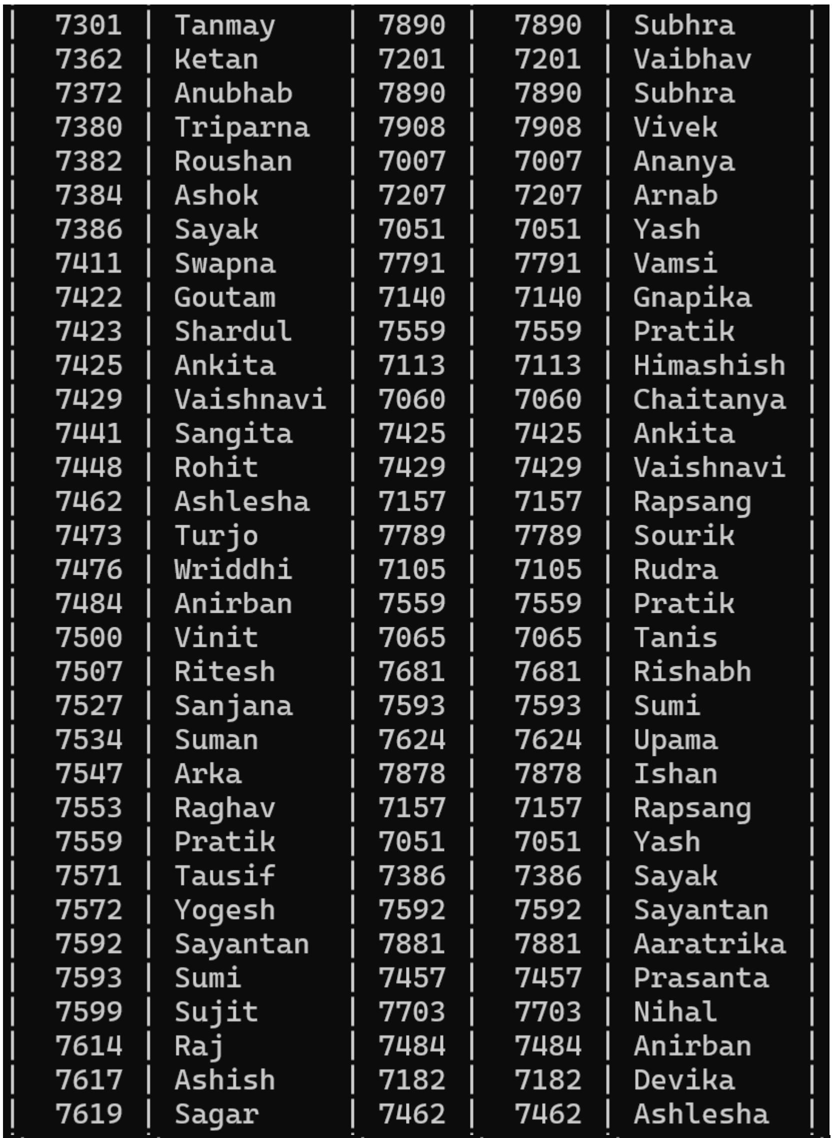
Query:

```
select e1.empno, e1.ename, e1.mgr,
  (select e2.empno from emp e2 where e2.empno = e1.mgr) as mgr_no,
  (select e2.ename from emp e2 where e2.empno = e1.mgr) as mgr_name
from emp e1
where e1.mgr is not null
having mgr_no is not null and mgr_name is not null;
```

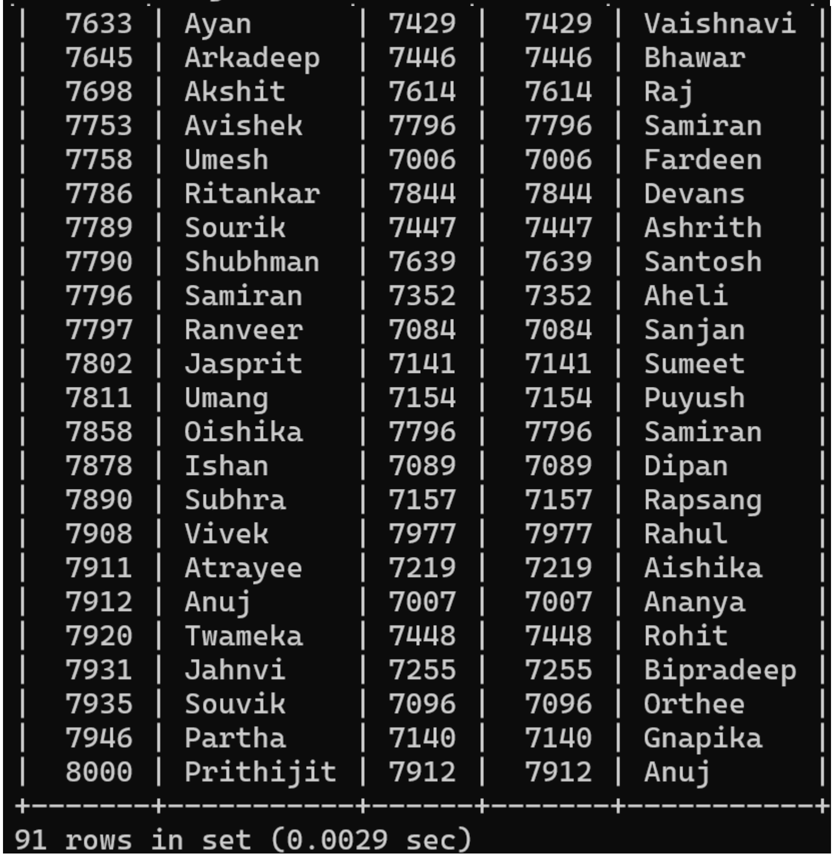
Output:



```
+-------+-----------+------+-------+-----------+
| empno | ename     | mgr  | empno | ename     |
+-------+-----------+------+-------+-----------+
|  7004 | Saikat    | 7116 |  7116 | Ayush     |
|  7006 | Fardeen   | 7067 |  7067 | Aditya    |
|  7007 | Ananya    | 7051 |  7051 | Yash      |
|  7013 | Sudip     | 7881 |  7881 | Aaratrika |
|  7036 | Sneha     | 7977 |  7977 | Rahul     |
|  7047 | Priya     | 7488 |  7488 | Joyabrata |
|  7060 | Chaitanya | 7422 |  7422 | Goutam    |
|  7064 | Prayas    | 7791 |  7791 | Vamsi     |
|  7077 | Nikhil    | 7231 |  7231 | Dinesh    |
|  7084 | Sanjan    | 7226 |  7226 | Tanuj     |
|  7085 | Nandini   | 7462 |  7462 | Ashlesha  |
|  7098 | Anudeep   | 7911 |  7911 | Atrayee   |
|  7106 | Sanju     | 7105 |  7105 | Rudra     |
|  7109 | John      | 7364 |  7364 | Raksha    |
|  7113 | Himashish | 7731 |  7731 | Sankhya   |
|  7116 | Ayush     | 7811 |  7811 | Umang     |
|  7128 | Sayantani | 7473 |  7473 | Turjo     |
|  7139 | Tarun     | 7249 |  7249 | Yuvraj    |
|  7140 | Gnapika   | 7219 |  7219 | Aishika   |
|  7141 | Sumeet    | 7105 |  7105 | Rudra     |
|  7154 | Puyush    | 7140 |  7140 | Gnapika   |
|  7157 | Rapsang   | 7425 |  7425 | Ankita    |
|  7175 | Rajkumar  | 7811 |  7811 | Umang     |
|  7201 | Vaibhav   | 7844 |  7844 | Devans    |
|  7202 | Rohan     | 7620 |  7620 | Neeharika |
|  7205 | Hamsa     | 7947 |  7947 | Soutrik   |
|  7207 | Arnab     | 7599 |  7599 | Sujit     |
|  7214 | Shreyas   | 7422 |  7422 | Goutam    |
|  7219 | Aishika   | 7753 |  7753 | Avishek   |
|  7227 | Sanmai    | 7797 |  7797 | Ranveer   |
|  7231 | Dinesh    | 7484 |  7484 | Anirban   |
|  7232 | Narendra  | 7061 |  7061 | Dipmay    |
|  7249 | Yuvraj    | 7084 |  7084 | Sanjan    |
|  7255 | Bipradeep | 7116 |  7116 | Ayush     |
|  7263 | Raman     | 7141 |  7141 | Sumeet    |
```



```
|  7301 | Tanmay     | 7890 |   7890 | Subhra     |
|  7362 | Ketan      | 7201 |   7201 | Vaibhav    |
|  7372 | Anubhab    | 7890 |   7890 | Subhra     |
|  7380 | Triparna   | 7908 |   7908 | Vivek      |
|  7382 | Roushan    | 7007 |   7007 | Ananya     |
|  7384 | Ashok      | 7207 |   7207 | Arnab      |
|  7386 | Sayak      | 7051 |   7051 | Yash       |
|  7411 | Swapna     | 7791 |   7791 | Vamsi      |
|  7422 | Goutam     | 7140 |   7140 | Gnapika    |
|  7423 | Shardul    | 7559 |   7559 | Pratik     |
|  7425 | Ankita     | 7113 |   7113 | Himashish  |
|  7429 | Vaishnavi  | 7060 |   7060 | Chaitanya  |
|  7441 | Sangita    | 7425 |   7425 | Ankita     |
|  7448 | Rohit      | 7429 |   7429 | Vaishnavi  |
|  7462 | Ashlesha   | 7157 |   7157 | Rapsang    |
|  7473 | Turjo      | 7789 |   7789 | Sourik     |
|  7476 | Wriddhi    | 7105 |   7105 | Rudra      |
|  7484 | Anirban    | 7559 |   7559 | Pratik     |
|  7500 | Vinit      | 7065 |   7065 | Tanis      |
|  7507 | Ritesh     | 7681 |   7681 | Rishabh    |
|  7527 | Sanjana    | 7593 |   7593 | Sumi       |
|  7534 | Suman      | 7624 |   7624 | Upama      |
|  7547 | Arka       | 7878 |   7878 | Ishan      |
|  7553 | Raghav     | 7157 |   7157 | Rapsang    |
|  7559 | Pratik     | 7051 |   7051 | Yash       |
|  7571 | Tausif     | 7386 |   7386 | Sayak      |
|  7572 | Yogesh     | 7592 |   7592 | Sayantan   |
|  7592 | Sayantan   | 7881 |   7881 | Aaratrika  |
|  7593 | Sumi       | 7457 |   7457 | Prasanta   |
|  7599 | Sujit      | 7703 |   7703 | Nihal      |
|  7614 | Raj        | 7484 |   7484 | Anirban    |
|  7617 | Ashish     | 7182 |   7182 | Devika     |
|  7619 | Sagar      | 7462 |   7462 | Ashlesha   |
```



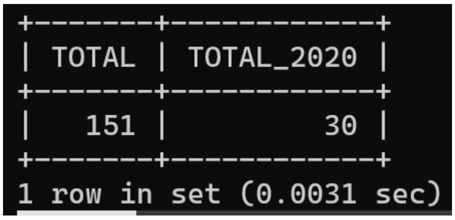
```
|  7633 | Ayan       | 7429 |   7429 | Vaishnavi  |
|  7645 | Arkadeep   | 7446 |   7446 | Bhawar     |
|  7698 | Akshit     | 7614 |   7614 | Raj        |
|  7753 | Avishek    | 7796 |   7796 | Samiran    |
|  7758 | Umesh      | 7006 |   7006 | Fardeen    |
|  7786 | Ritankar   | 7844 |   7844 | Devans     |
|  7789 | Sourik     | 7447 |   7447 | Ashrith    |
|  7790 | Shubhman   | 7639 |   7639 | Santosh    |
|  7796 | Samiran    | 7352 |   7352 | Aheli      |
|  7797 | Ranveer    | 7084 |   7084 | Sanjan     |
|  7802 | Jasprit    | 7141 |   7141 | Sumeet     |
|  7811 | Umang      | 7154 |   7154 | Puyush     |
|  7858 | Oishika    | 7796 |   7796 | Samiran    |
|  7878 | Ishan      | 7089 |   7089 | Dipan      |
|  7890 | Subhra     | 7157 |   7157 | Rapsang    |
|  7908 | Vivek      | 7977 |   7977 | Rahul      |
|  7911 | Atrayee    | 7219 |   7219 | Aishika    |
|  7912 | Anuj       | 7007 |   7007 | Ananya     |
|  7920 | Twameka    | 7448 |   7448 | Rohit      |
|  7931 | Jahnvi     | 7255 |   7255 | Bipradeep  |
|  7935 | Souvik     | 7096 |   7096 | Orthee     |
|  7946 | Partha     | 7140 |   7140 | Gnapika    |
|  8000 | Prithijit  | 7912 |   7912 | Anuj       |
+-------+------------+------+--------+------------+
91 rows in set (0.0029 sec)
```

3. **Create a query that will display the total number of employees and the total number of employees who were hired only in 2020. Give the column headings as “TOTAL” and “TOTAL_2020” respectively.**

Query:

select count(*) as TOTAL, count(case when year(hiredate) = 2020 then 1 else null end) as TOTAL_2020
from emp;

Output:



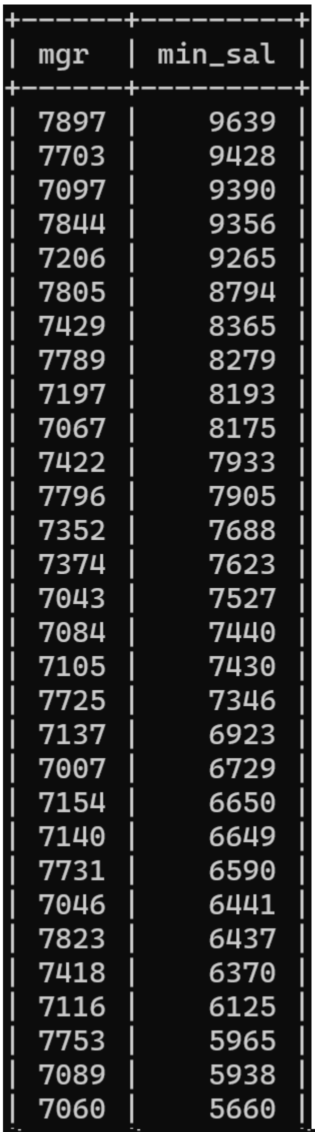
```
+-------+------------+
| TOTAL | TOTAL_2020 |
+-------+------------+
|   151 |         30 |
+-------+------------+
1 row in set (0.0031 sec)
```

4. **Display the manager number and the salary of the lowest paid employee under that manager. Exclude anyone whose manager is not known. Exclude any group where the minimum salaries less than 1000. Sort the output in descending order of salary.**

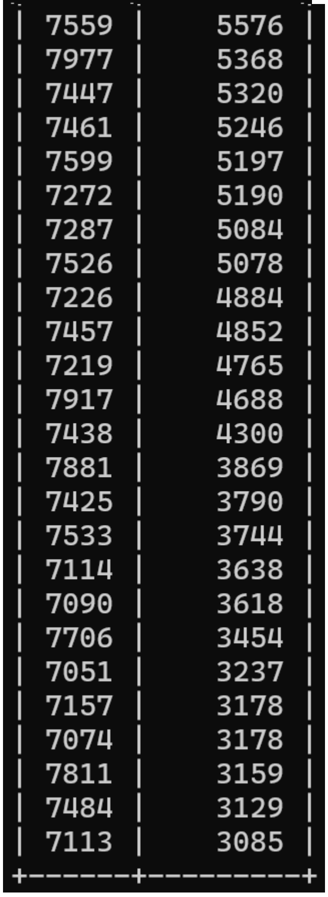
Query:

```
select e1.empno as "manager number",
    (select min(e2.sal) from emp e2 where e2.mgr = e1.empno) as min_sal
from emp e1
where e1.empno in (select distinct mgr from emp where mgr is not null)
group by e1.empno
having (select min(e2.sal) from emp e2 where e2.mgr = e1.empno) >= 1000
order by min_sal desc;
```

Output:



```
+------+---------+
| mgr  | min_sal |
+------+---------+
| 7897 |    9639 |
| 7703 |    9428 |
| 7097 |    9390 |
| 7844 |    9356 |
| 7206 |    9265 |
| 7805 |    8794 |
| 7429 |    8365 |
| 7789 |    8279 |
| 7197 |    8193 |
| 7067 |    8175 |
| 7422 |    7933 |
| 7796 |    7905 |
| 7352 |    7688 |
| 7374 |    7623 |
| 7043 |    7527 |
| 7084 |    7440 |
| 7105 |    7430 |
| 7725 |    7346 |
| 7137 |    6923 |
| 7007 |    6729 |
| 7154 |    6650 |
| 7140 |    6649 |
| 7731 |    6590 |
| 7046 |    6441 |
| 7823 |    6437 |
| 7418 |    6370 |
| 7116 |    6125 |
| 7753 |    5965 |
| 7089 |    5938 |
| 7060 |    5660 |
```



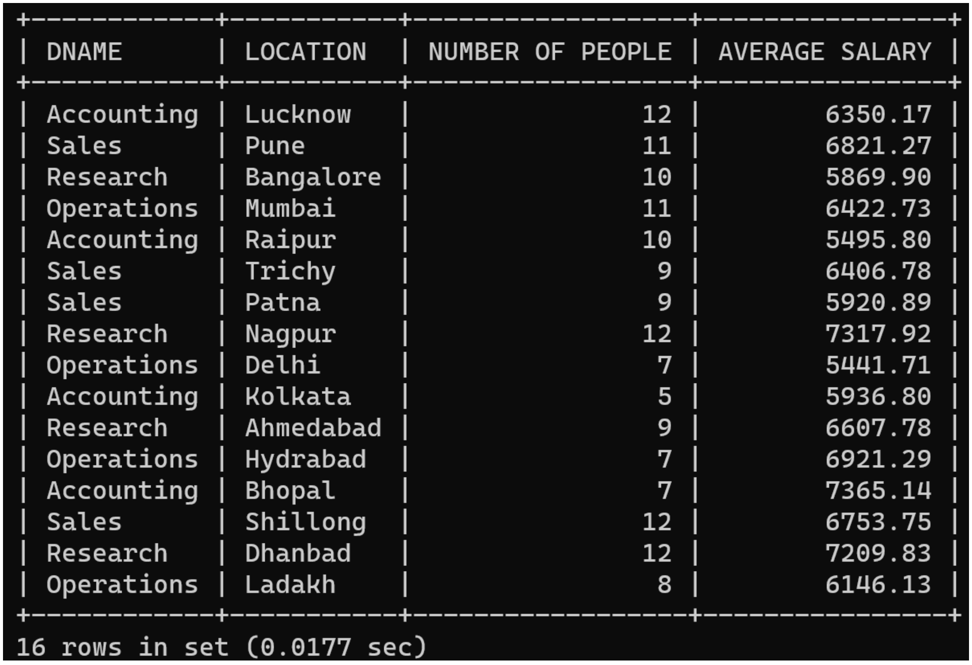
```
| 7559 |    5576 |
| 7977 |    5368 |
| 7447 |    5320 |
| 7461 |    5246 |
| 7599 |    5197 |
| 7272 |    5190 |
| 7287 |    5084 |
| 7526 |    5078 |
| 7226 |    4884 |
| 7457 |    4852 |
| 7219 |    4765 |
| 7917 |    4688 |
| 7438 |    4300 |
| 7881 |    3869 |
| 7425 |    3790 |
| 7533 |    3744 |
| 7114 |    3638 |
| 7090 |    3618 |
| 7706 |    3454 |
| 7051 |    3237 |
| 7157 |    3178 |
| 7074 |    3178 |
| 7811 |    3159 |
| 7484 |    3129 |
| 7113 |    3085 |
+------+---------+
```

5. **Assume that there are some departments where no employee is assigned. Now, write a query to display the department name, location name, number of employees, and the average salary for all the employees in that department. Label the columns as “DNAME”, “LOCATION”, “NUMBER OF PEOPLE”, and “AVERAGE SALARY” respectively. Round the averge salary to two decimal places. The outcome of the query must include the details of the departments where no employee is assigned and in that case the “AVERAGE SALARY” for that department is to be displayed as 0(zero).**

Query:

```
select d.dname as "DNAME",
    d.loc as "LOCATION",
    (select count(e.empno) from emp e where e.deptno = d.deptno) as
"NUMBER OF PEOPLE",
    (select round(avg(e.sal), 2) from emp e where e.deptno = d.deptno) as
"AVERAGE SALARY"
from dept d
union all
select d.dname as "DNAME",
    d.loc as "LOCATION",
    0 as "NUMBER OF PEOPLE",
    0.00 as "AVERAGE SALARY"
from dept d
where d.deptno not in (select distinct deptno from emp)
order by "DNAME";
```

Output:



```
+------------+-----------+------------------+----------------+
| DNAME      | LOCATION  | NUMBER OF PEOPLE | AVERAGE SALARY |
+------------+-----------+------------------+----------------+
| Accounting | Lucknow   |               12 |        6350.17 |
| Sales      | Pune      |               11 |        6821.27 |
| Research   | Bangalore |               10 |        5869.90 |
| Operations | Mumbai    |               11 |        6422.73 |
| Accounting | Raipur    |               10 |        5495.80 |
| Sales      | Trichy    |                9 |        6406.78 |
| Sales      | Patna     |                9 |        5920.89 |
| Research   | Nagpur    |               12 |        7317.92 |
| Operations | Delhi     |                7 |        5441.71 |
| Accounting | Kolkata   |                5 |        5936.80 |
| Research   | Ahmedabad |                9 |        6607.78 |
| Operations | Hydrabad  |                7 |        6921.29 |
| Accounting | Bhopal    |                7 |        7365.14 |
| Sales      | Shillong  |               12 |        6753.75 |
| Research   | Dhanbad   |               12 |        7209.83 |
| Operations | Ladakh    |                8 |        6146.13 |
+------------+-----------+------------------+----------------+
16 rows in set (0.0177 sec)
```