Sheet 4

NAME: ARNAB SEN
EN.NO: 510519006 (Gx)
SUBJECT: DBMS Lab

Assignment No: 8

Write the SQL commands to create the following relational schemas:

Customer (cust_id, cust_name, annual_revenue, cust_type)

Cust_id must be between 100 and 10,000.

Cust_type must be 'MANUFACTURER', 'WHOLESALER', or 'RETAILER'.

query:

```
CREATE TABLE customer(
    cust_id numeric(6) PRIMARY KEY CHECK(
        (cust_id > 100)
        AND (cust_id < 10000)
    ),
    annual_income numeric(8),
    cust_namen varchar(20),
    cust_type varchar(24) NOT NULL CHECK(
        cust_type IN('MANUFACTURER', 'WHOLESALER', 'RETAILER')
    )
);
```

```
lab2=> \d customer
                      Table "public.customer"
    Column     |         Type          | Collation | Nullable | Default
---------------+-----------------------+-----------+----------+---------
 cust_id       | numeric(6,0)          |           | not null |
 annual_income | numeric(8,0)          |           |          |
 cust_namen    | character varying(20) |           |          |
 cust_type     | character varying(24) |           | not null |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (cust_id)
Check constraints:
    "customer_cust_id_check" CHECK (cust_id > 100::numeric AND cust_id < 10000::numeric)
    "customer_cust_type_check" CHECK (cust_type::text = ANY (ARRAY['MANUFACTURER'::character varying
, 'WHOLESALER'::character varying, 'RETAILER'::character varying]::text[]))
```

Truck (truck_no, driver_name)

query:

```
CREATE TABLE truck(
    truck_no VARCHAR(10) PRIMARY KEY,
    driver_name VARCHAR (20)
);
```

```
lab2=> \d truck
                       Table "public.truck"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 truck_no    | character varying(10) |           | not null |
 driver_name | character varying(20) |           |          |
Indexes:
    "truck_pkey" PRIMARY KEY, btree (truck_no)
Referenced by:
    TABLE "shipment" CONSTRAINT "shipment_truck_no_fkey" FOREIGN KEY (truck_no) REFERENCES truck(tru
ck_no) ON DELETE SET NULL
```

City (city_name, population)

query:

```
CREATE TABLE city(
    city_name VARCHAR(20) PRIMARY KEY,
    population NUMERIC(10)
);
```

```
lab2=> \d city
                      Table "public.city"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 city_name  | character varying(20) |           | not null |
 population | numeric(10,0)         |           |          |
Indexes:
    "city_pkey" PRIMARY KEY, btree (city_name)
Referenced by:
    TABLE "shipment" CONSTRAINT "shipment_destination_fkey" FOREIGN KEY (destination) REFERENCES cit
y(city_name)
```

Shipment (shipment_no, cust_id, weight, truck_no, destination, ship_date)
Foreign keys: cust_id references customer on delete cascade, truck_no references truck on delete set null, destination references city. Weight must be under 1000.

query:

```
CREATE TABLE shipment(
    shipment_no VARCHAR(6),
    cust_id NUMERIC(6) REFERENCES customer(cust_id) ON DELETE CASCADE,
    weight NUMERIC(4) CHECK(weight < 1000),
    truck_no VARCHAR(10) REFERENCES truck(truck_no) ON DELETE SET NULL,
    destination VARCHAR(20) REFERENCES city(city_name),
    ship_date DATE,
    PRIMARY KEY(shipment_no, cust_id)
);
```

```
lab2=> \d shipment
                    Table "public.shipment"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 shipment_no | character varying(6)  |           | not null |
 cust_id     | numeric(6,0)          |           | not null |
 weight      | numeric(4,0)          |           |          |
 truck_no    | character varying(10) |           |          |
 destination | character varying(20) |           |          |
 ship_date   | date                  |           |          |
Indexes:
    "shipment_pkey" PRIMARY KEY, btree (shipment_no, cust_id)
Check constraints:
    "shipment_weight_check" CHECK (weight < 1000::numeric)
Foreign-key constraints:
    "shipment_cust_id_fkey" FOREIGN KEY (cust_id) REFERENCES customer(cust_id) ON DELETE CASCADE
    "shipment_destination_fkey" FOREIGN KEY (destination) REFERENCES city(city_name)
    "shipment_truck_no_fkey" FOREIGN KEY (truck_no) REFERENCES truck(truck_no) ON DELETE SET NULL
```

```
lab2=> \d
         List of relations
 Schema |   Name   | Type  | Owner
--------+----------+-------+-------
 public | city     | table | arnab
 public | customer | table | arnab
 public | shipment | table | arnab
 public | truck    | table | arnab
(4 rows)
```

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

query:

```
SELECT
    DISTINCT cust_name
FROM
    shipment s,
    customer c
```

```
WHERE
    s.cust_id = c.cust_id
    AND destination IN ('Chennai', 'Kolkata', 'Mumbai');
```

```
lab2=> SELECT
    DISTINCT cust_name
FROM
    shipment s,
    customer c
WHERE
    s.cust_id = c.cust_id
    AND destination IN ('Chennai', 'Kolkata', 'Mumbai');
 cust_name
-----------
 Anay
 Dhruv
 Divij
 Drishya
 Faiyaz
 Jayesh
 Pihu
 Shanaya
 Trisha
(9 rows)
```

2. List the names of the driver who have delivered shipments weighing over 200 pounds.

query:

```
SELECT
    DISTINCT driver_name
```

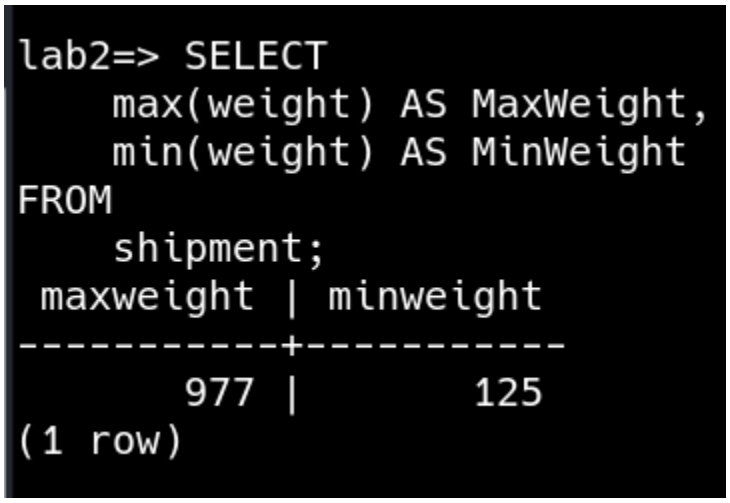
```
FROM
    shipment s,
    truck t
WHERE
    s.truck_no = t.truck_no
    AND s.weight > 200;
```

```
lab2=> SELECT
    DISTINCT driver_name
FROM
    shipment s,
    truck t
WHERE
    s.truck_no = t.truck_no
    AND s.weight > 200;
 driver_name
-------------
 Kiara
 Khushi
 Riaan
 Anika
 Badal
 Drishya
 Fateh
 Purab
 Sumer
 Chirag
(10 rows)
```

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 MaxWeight,
    min(weight) AS MinWeight
FROM
    shipment;
```



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

query:

```
SELECT
    cust_id,
    avg(weight) AS AverageWeight
FROM
    shipment
GROUP BY
    cust_id
ORDER BY
    cust_id;
```

```
lab2=> SELECT
    cust_id,
    avg(weight) AS AverageWeight
FROM
    shipment
GROUP BY
    cust_id
ORDER BY
    cust_id;
 cust_id |    averageweight
---------+---------------------
    1501 | 354.3333333333333
    1502 | 319.8000000000000
    1503 | 629.6666666666667
    1504 | 608.2000000000000
    1505 | 662.0909090909090
    1506 | 504.2000000000000
    1507 | 706.3333333333333
    1508 | 428.5000000000000
    1509 | 504.2500000000000
    1510 | 742.0000000000000
(10 rows)
```

5. List the names and populations of cities that have received a shipment weighing over 100 pounds.

query:

```
SELECT
    city_name,
    population
FROM
    shipment s,
    city c
```

```
WHERE
    s.destination = c.city_name
    AND weight > 100
GROUP BY
    city_name;
```

```
lab2=> SELECT
    city_name,
    population
FROM
    shipment s,
    city c
WHERE
    s.destination = c.city_name
    AND weight > 100
GROUP BY
    city_name;
 city_name | population
-----------+------------
 Mumbai    |    7200000
 Delhi     |   10000000
 Chennai   |    8000000
 Hyderabad |    6000000
 Bangalore |    5100000
 Kolkata   |    5000000
(6 rows)
```

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

query:

```
SELECT
    city_name
```

```
FROM
    shipment s,
    city c
WHERE
    s.destination = c.city_name
GROUP BY
    city_name
HAVING
    count(DISTINCT cust_id) = (
        SELECT
            count(*)
        FROM
            customer
    );
```

```
lab2=> SELECT
    city_name
FROM
    shipment s,
    city c
WHERE
    s.destination = c.city_name
GROUP BY
    city_name
HAVING
    count(DISTINCT cust_id) = (
        SELECT
            count(*)
        FROM
            customer
    );
 city_name
-----------
 Mumbai
(1 row)
```

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

query:

```
SELECT
    city_name,
    max(weight)
FROM
    city c,
    shipment s
WHERE
    c.city_name = s.destination
GROUP BY
    city_name;
```

```
lab2=> SELECT
    city_name,
    max(weight)
FROM
    city c,
    shipment s
WHERE
    c.city_name = s.destination
GROUP BY
    city_name;
 city_name | max
-----------+-----
 Hyderabad | 920
 Bangalore | 903
 Mumbai    | 886
 Delhi     | 714
 Kolkata   | 977
 Chennai   | 930
(6 rows)
```

8. List the name and annual revenue of customers whose shipments have been delivered by truck driver 'Kiara'.

query:

```
SELECT
    cust_name,
    annual_revenue
FROM
    customer c,
    shipment s,
    truck t
WHERE
```

```
    c.cust_id = s.cust_id
    AND s.truck_no = t.truck_no
    AND driver_name = 'Kiara';
```

```
lab2=> SELECT
    cust_name,
    annual_revenue
FROM
    customer c,
    shipment s,
    truck t
WHERE
    c.cust_id = s.cust_id
    AND s.truck_no = t.truck_no
    AND driver_name = 'Kiara';
 cust_name | annual_revenue
-----------+----------------
 Divij     |        5928153
 Divij     |        3211435
 Divij     |        5928153
 Faiyaz    |        9204049
 Shanaya   |        7812942
 Pihu      |        2066160
 Divij     |        5928153
(7 rows)
```

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

query:

```
SELECT
    t.truck_no,
    t.driver_name
```

```
FROM
    shipment s,
    truck t
WHERE
    s.truck_no = t.truck_no
GROUP BY
    t.truck_no
HAVING
    count(DISTINCT(destination)) = (
        SELECT
            count(*)
        FROM
            city
    );
```

```
lab2=> SELECT
    t.truck_no,
    t.driver_name
FROM
    shipment s,
    truck t
WHERE
    s.truck_no = t.truck_no
GROUP BY
    t.truck_no
HAVING
    count(DISTINCT(destination)) = (
        SELECT
            count(*)
        FROM
            city
    );
 truck_no | driver_name
----------+-------------
    18002 | Chirag
(1 row)
```

10. For each city, with a population of 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;
```

```
lab2=> SELECT
    city_name,
    min(weight)
FROM
    shipment,
    city
WHERE
    destination = city_name
    AND population > 1000000
GROUP BY
    city_name;
 city_name | min
-----------+-----
 Kolkata   | 175
 Hyderabad | 198
 Bangalore | 268
 Mumbai    | 132
 Delhi     | 125
 Chennai   | 269
(6 rows)
```

Assignment No: 9

Write SQL commands to create the following tables as well as to insert sufficient number of values in the tables:

EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)

EMPNO must be between 7000 and 8000.

ENAME must not exceed 10 characters.

JOB must be in (‘Clerk’,‘Salesman’,‘Manager’,‘Analyst’,‘President’).

MGR is the manager’s EMPNO.

COMM must be under 1500 and defaults to 0.

```
CREATE TABLE EMP(
    EMPNO numeric(10) PRIMARY KEY CHECK (
        EMPNO BETWEEN 7000
        AND 8000
    ),
    ENAME varchar(10),
    JOB varchar(10) CHECK (
        JOB IN(
            'President',
            'Manager',
            'Clerk',
            'Salesman',
            'Analyst'
        )
    ),
    MGR numeric(10),
    HIREDATE date,
    SAL numeric(10),
    COMM numeric(5) DEFAULT 0 CHECK(COMM < 1500),
    DEPTNO varchar(5) REFERENCES DEPT(DEPTNO)
);
```

DEPT (DEPTNO, DNAME, LOC)

DEPTNO must start with ‘D’.

DNAME must be ‘Accounting’ or ‘Sales’ or ‘Research’ or ‘Operations’.

```
CREATE TABLE DEPT(
    DEPTNO varchar(5) PRIMARY KEY CHECK (DEPTNO LIKE 'D%'),
    DNAME varchar(10) CHECK (
        DNAME IN ('Accounting', 'Sales', 'Research', 'Operations')
    ),
    LOC varchar(10)
);
```

1. Display the difference between the highest and lowest salaries of each department in descending order. Label the column as “Difference”.

query:

```
SELECT
    DNAME,
    max(SAL) - min(SAL) AS Difference
FROM
    DEPT d,
    EMP e
WHERE
    d.DEPTNO = e.DEPTNO
GROUP BY
    D.DEPTNO
ORDER BY
    Difference DESC;
```

```
lab3=> SELECT
    DNAME,
    max(SAL) - min(SAL) AS SalaryDifference
FROM
    DEPT d,
    EMP e
WHERE
    d.DEPTNO = e.DEPTNO
GROUP BY
    D.DEPTNO
ORDER BY
    SalaryDifference DESC;
   dname    | salarydifference
------------+------------------
 Research   |           751422
 Accounting |           700851
 Sales      |           619354
 Operations |           485127
(4 rows)
```

2. List all the employees' employee numbers and names along with their immediate managers' employee numbers and names.

query:

```
SELECT
    a.EMPNO AS employee_id,
    a.ENAME AS employee_name,
    b.EMPNO AS manager_id,
    b.ENAME AS manager_name
FROM
    EMP a,
```

```
    EMP b
WHERE
    a.MGR = b.EMPNO;
```

employee_id	employee_name	manager_id	manager_name
7001	Siya	7012	Anya
7002	Nitara	7003	Taran
7003	Taran	7021	Drishya
7004	Ryan	7013	Myra
7005	Elakshi	7020	Shaan
7006	Tiya	7017	Nitya
7007	Anay	7019	Zoya
7008	Emir	7018	Hridaan
7009	Darshit	7011	Manjari
7010	Riaan	7006	Tiya
7011	Manjari	7029	Emir
7012	Anya	7020	Shaan
7013	Myra	7001	Siya
7014	Samarth	7020	Shaan
7015	Pranay	7010	Riaan
7016	Anya	7019	Zoya
7017	Nitya	7012	Anya
7018	Hridaan	7028	Akarsh
7019	Zoya	7012	Anya
7020	Shaan	7010	Riaan
7021	Drishya	7019	Zoya
7022	Arhaan	7003	Taran
7023	Jivika	7006	Tiya
7024	Renee	7015	Pranay
7025	Uthkarsh	7021	Drishya
7026	Stuvan	7024	Renee
7027	Neysa	7004	Ryan
7028	Akarsh	7018	Hridaan
7029	Emir	7008	Emir
7030	Tejas	7015	Pranay

(30 rows)

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
    TOTAL,
    TOTAL_2020
FROM
    (
        SELECT
            count(*) AS TOTAL
        FROM
            EMP
    ) AS hd1,
    (
        SELECT
            count(*) AS TOTAL_2020
        FROM
            EMP
        WHERE
            EXTRACT(
                YEAR
                FROM
                    HIREDATE
            ) = 2020
    ) AS hd2;
```

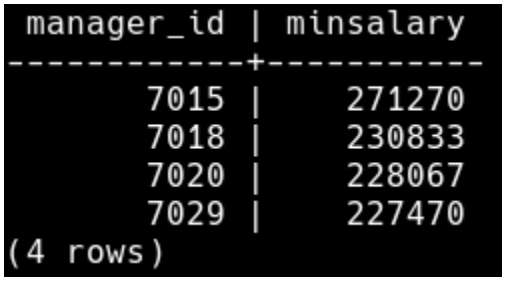
```
lab3=> SELECT
    TOTAL,
    TOTAL_2020
FROM
    (
        SELECT
            count(*) AS TOTAL
        FROM
            EMP
    ) AS hd1,
    (
        SELECT
            count(*) AS TOTAL_2020
        FROM
            EMP
        WHERE
            EXTRACT(
                YEAR
                FROM
                    HIREDATE
            ) = 2020
    ) AS hd2;
 total | total_2020
-------+------------
    30 |          6
(1 row)
```

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 salary is less than 1000. Sort the output in descending order of salary.

query:

```
SELECT
    MGR AS manager_id,
    min(SAL) AS minSalary
```

```
FROM
    EMP
GROUP BY
    MGR
HAVING
    MGR IS NOT NULL
    AND min(SAL) < 300000
ORDER BY
    minSalary DESC;
```



```
 manager_id | minsalary
------------+-----------
       7015 |    271270
       7018 |    230833
       7020 |    228067
       7029 |    227470
(4 rows)
```

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 average 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
```

```
    DNAME AS "DNAME",
    LOC AS "LOCATION",
    count(*) AS "NUMBER OF PEOPLE",
    coalesce(round(avg(SAL), 2), 0.0) AS "AVERAGE SALARY"
FROM
    DEPT
    LEFT JOIN EMP ON (EMP.DEPTNO = DEPT.DEPTNO)
GROUP BY
    DNAME,
    LOC;
```

```
   DNAME    | LOCATION  | NUMBER OF PEOPLE | AVERAGE SALARY
------------+-----------+------------------+----------------
 Accounting | Bangalore |                7 |      486607.00
 Research   | Delhi     |                9 |      531843.22
 Operations | Kolkata   |                8 |      581090.88
 Sales      | Mumbai    |                6 |      512094.50
(4 rows)
```