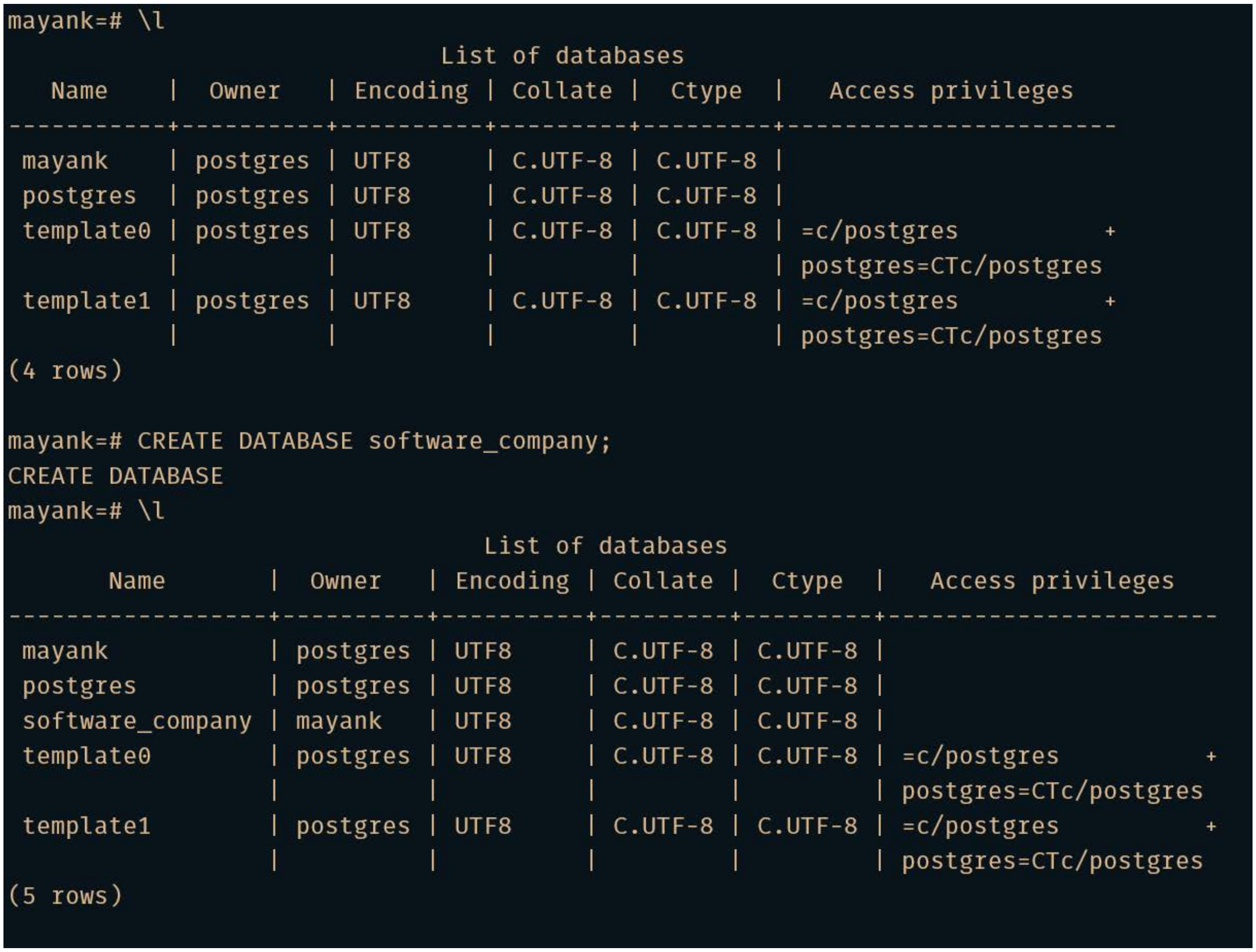
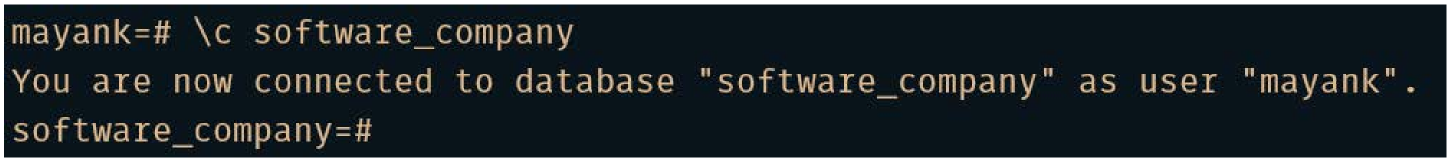
CS3120 Data Base Management Systems Laboratory
Lab 1

Mayank Singla
111901030

Q1. Create a Software Company database with name <database name >.



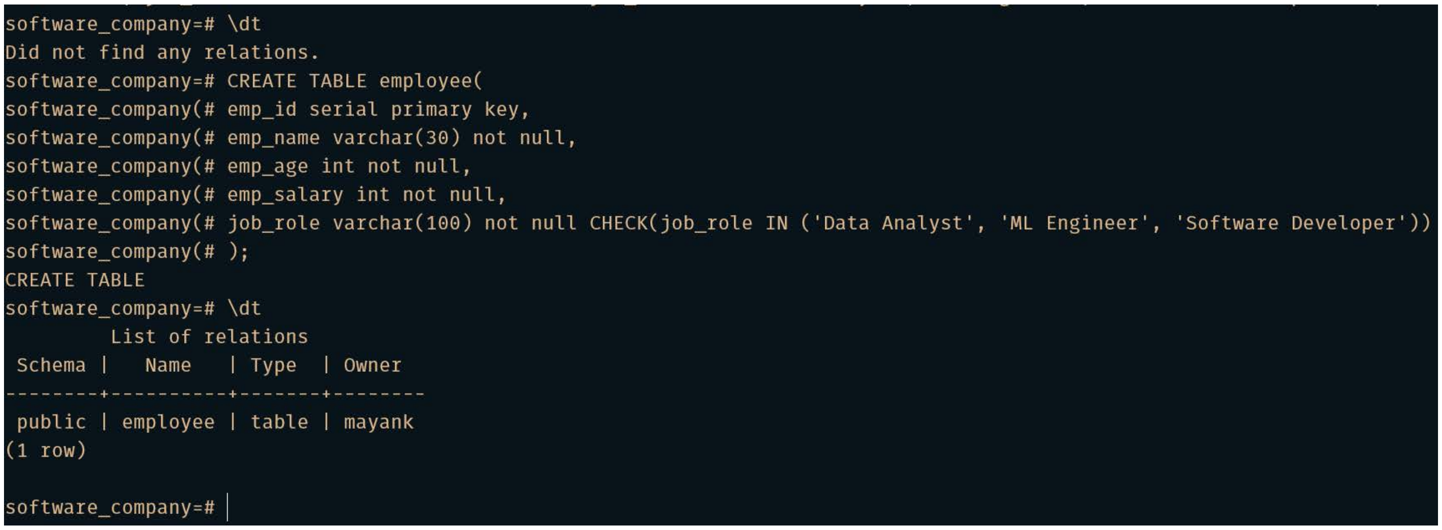
```
mayank=# \l
                              List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 mayank    | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
(4 rows)

mayank=# CREATE DATABASE software_company;
CREATE DATABASE
mayank=# \l
                                  List of databases
       Name       |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
------------------+----------+----------+---------+---------+-----------------------
 mayank           | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 postgres         | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 software_company | mayank   | UTF8     | C.UTF-8 | C.UTF-8 |
 template0        | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
                  |          |          |         |         | postgres=CTc/postgres
 template1        | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
                  |          |          |         |         | postgres=CTc/postgres
(5 rows)
```



```
mayank=# \c software_company
You are now connected to database "software_company" as user "mayank".
software_company=#
```

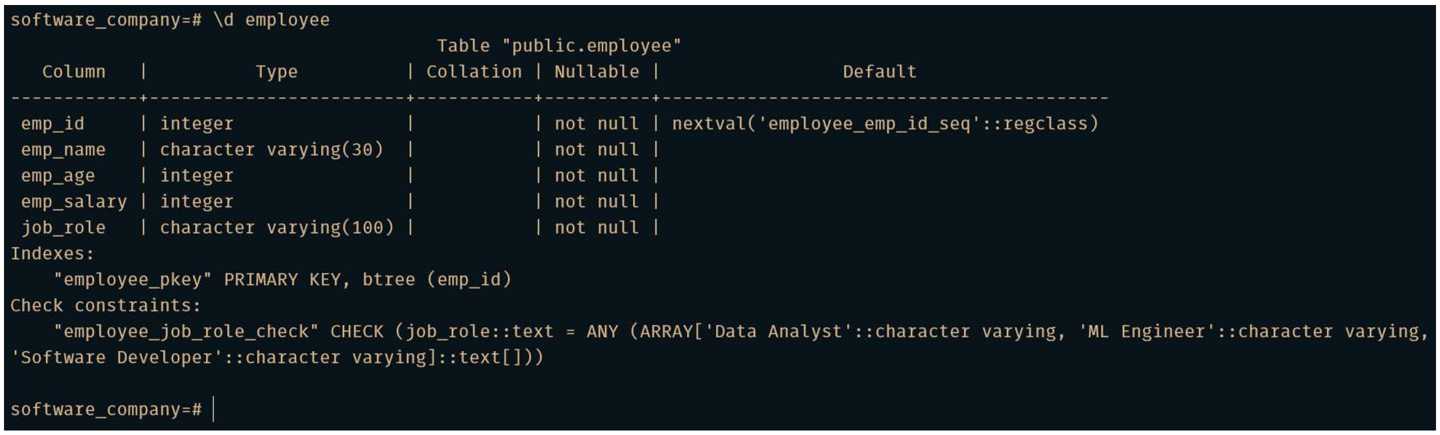
Creating the Database using CREATE DATABASE <db_name> command and then connecting to that database.

Q2. Create an employee table with 5 columns(emp id,emp name,emp age,emp salary,job role) and 10 records. Choose a unique column as a primary key and also other constraints as per your understanding.
Note: Job roles available(”Data Analyst”,”ML Engineer”,”Software Developer”).



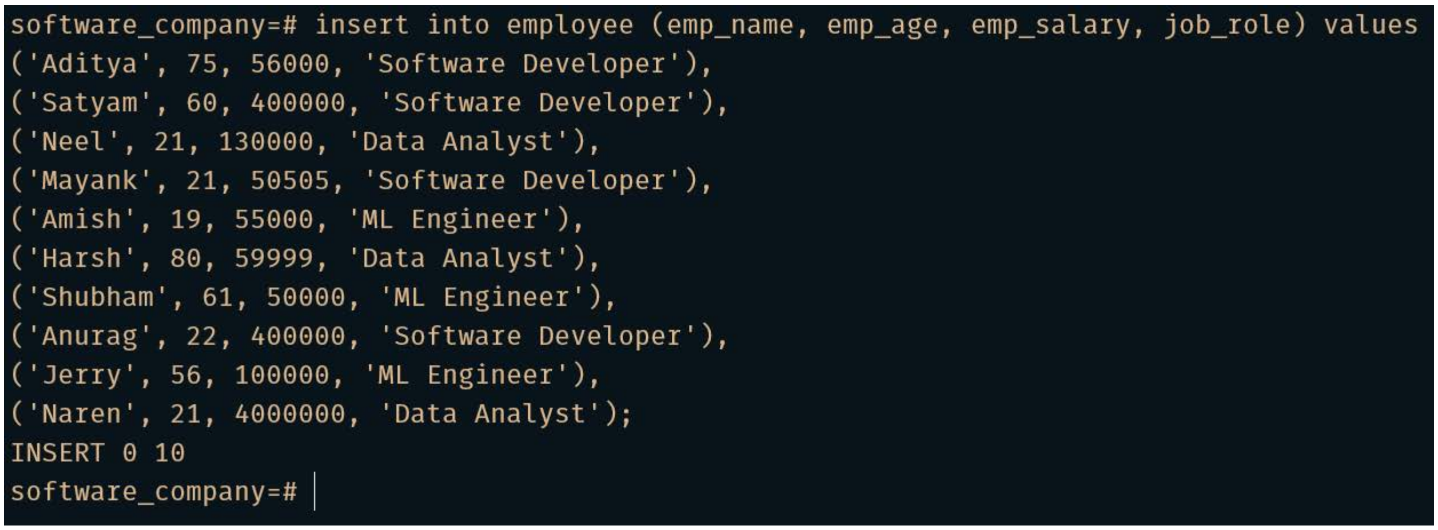
```
software_company=# \dt
Did not find any relations.
software_company=# CREATE TABLE employee(
software_company(# emp_id serial primary key,
software_company(# emp_name varchar(30) not null,
software_company(# emp_age int not null,
software_company(# emp_salary int not null,
software_company(# job_role varchar(100) not null CHECK(job_role IN ('Data Analyst', 'ML Engineer', 'Software Developer'))
software_company(# );
CREATE TABLE
software_company=# \dt
         List of relations
 Schema |   Name   | Type  | Owner
--------+----------+-------+--------
 public | employee | table | mayank
(1 row)

software_company=#
```

Creating the table **empoyee** with the fields mentioned in the question and giving appropriate datatype.
I am putting all the fields as not null and selecting emp_id as my *Primary Key*.
For job_role I have put a CHECK constraint to make sure that it's value is only possible out of the given 3 values.



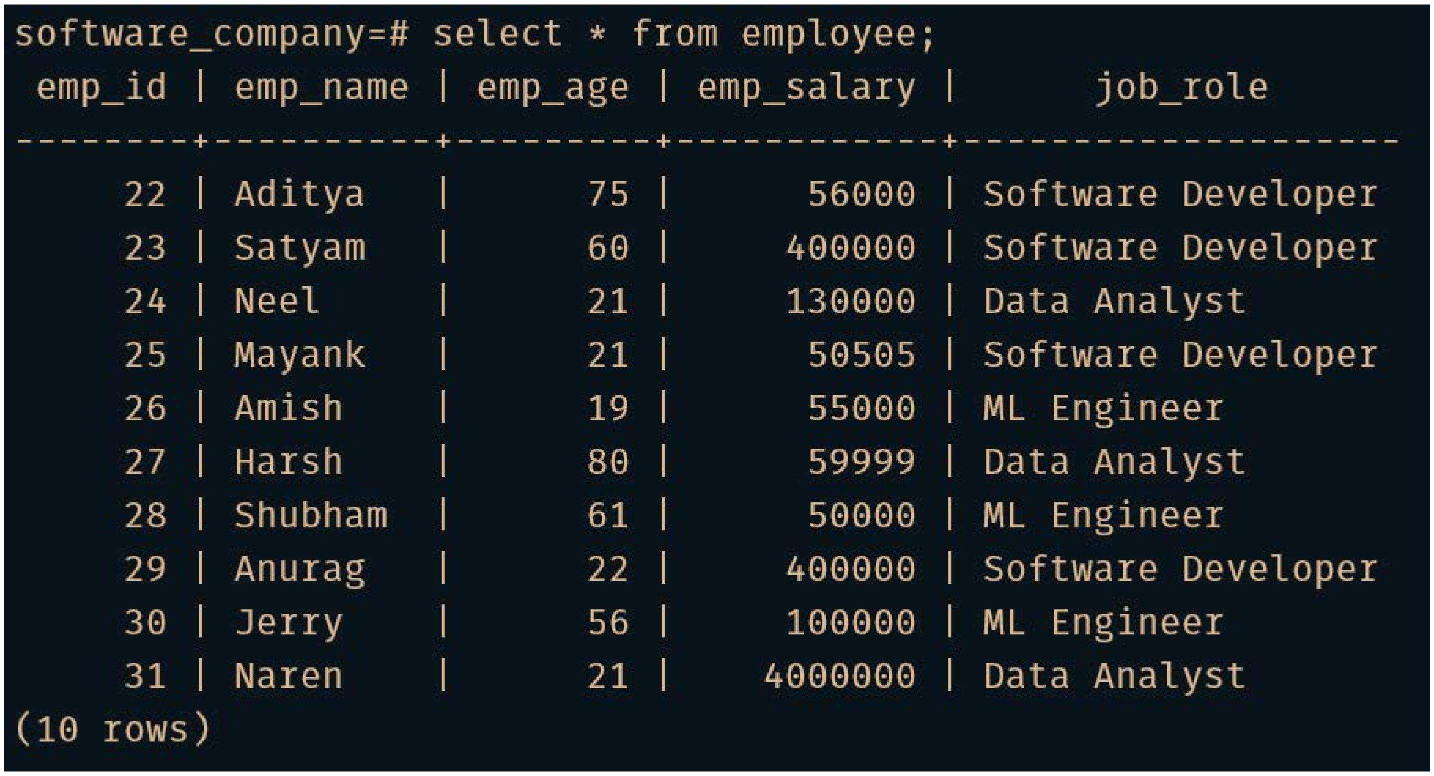
```
software_company=# \d employee
                                    Table "public.employee"
   Column   |          Type          | Collation | Nullable |                 Default
------------+------------------------+-----------+----------+------------------------------------------
 emp_id     | integer                |           | not null | nextval('employee_emp_id_seq'::regclass)
 emp_name   | character varying(30)  |           | not null |
 emp_age    | integer                |           | not null |
 emp_salary | integer                |           | not null |
 job_role   | character varying(100) |           | not null |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (emp_id)
Check constraints:
    "employee_job_role_check" CHECK (job_role::text = ANY (ARRAY['Data Analyst'::character varying, 'ML Engineer'::character varying,
'Software Developer'::character varying]::text[]))

software_company=#
```

The structure of the created table.



```
software_company=# insert into employee (emp_name, emp_age, emp_salary, job_role) values
('Aditya', 75, 56000, 'Software Developer'),
('Satyam', 60, 400000, 'Software Developer'),
('Neel', 21, 130000, 'Data Analyst'),
('Mayank', 21, 50505, 'Software Developer'),
('Amish', 19, 55000, 'ML Engineer'),
('Harsh', 80, 59999, 'Data Analyst'),
('Shubham', 61, 50000, 'ML Engineer'),
('Anurag', 22, 400000, 'Software Developer'),
('Jerry', 56, 100000, 'ML Engineer'),
('Naren', 21, 4000000, 'Data Analyst');
INSERT 0 10
software_company=#
```

Inserting 10 records in the table with random data and satisfying all the constraints.



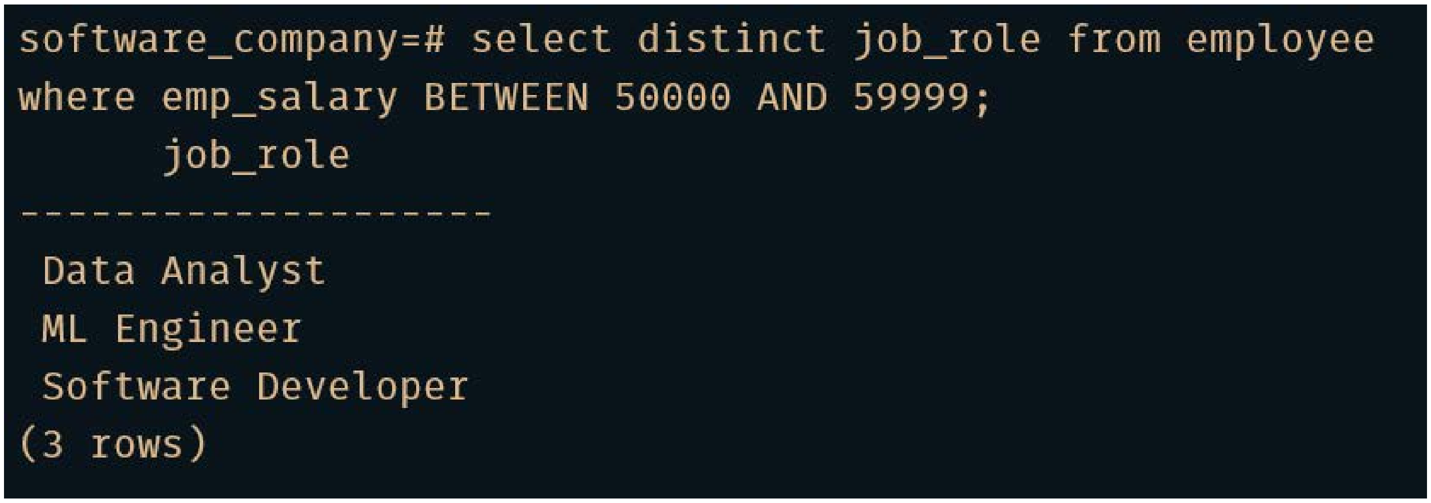
```
software_company=# select * from employee;
 emp_id | emp_name | emp_age | emp_salary |      job_role
--------+----------+---------+------------+--------------------
     22 | Aditya   |      75 |      56000 | Software Developer
     23 | Satyam   |      60 |     400000 | Software Developer
     24 | Neel     |      21 |     130000 | Data Analyst
     25 | Mayank   |      21 |      50505 | Software Developer
     26 | Amish    |      19 |      55000 | ML Engineer
     27 | Harsh    |      80 |      59999 | Data Analyst
     28 | Shubham  |      61 |      50000 | ML Engineer
     29 | Anurag   |      22 |     400000 | Software Developer
     30 | Jerry    |      56 |     100000 | ML Engineer
     31 | Naren    |      21 |    4000000 | Data Analyst
(10 rows)
```

Checking the contents of the table.

NOTE: *I have tried adding invalid data before just to check whether my constraints are working or not, that is why the emp_id is not starting from 1.*

Q3. Retrieve distinct job roles of the employees whose

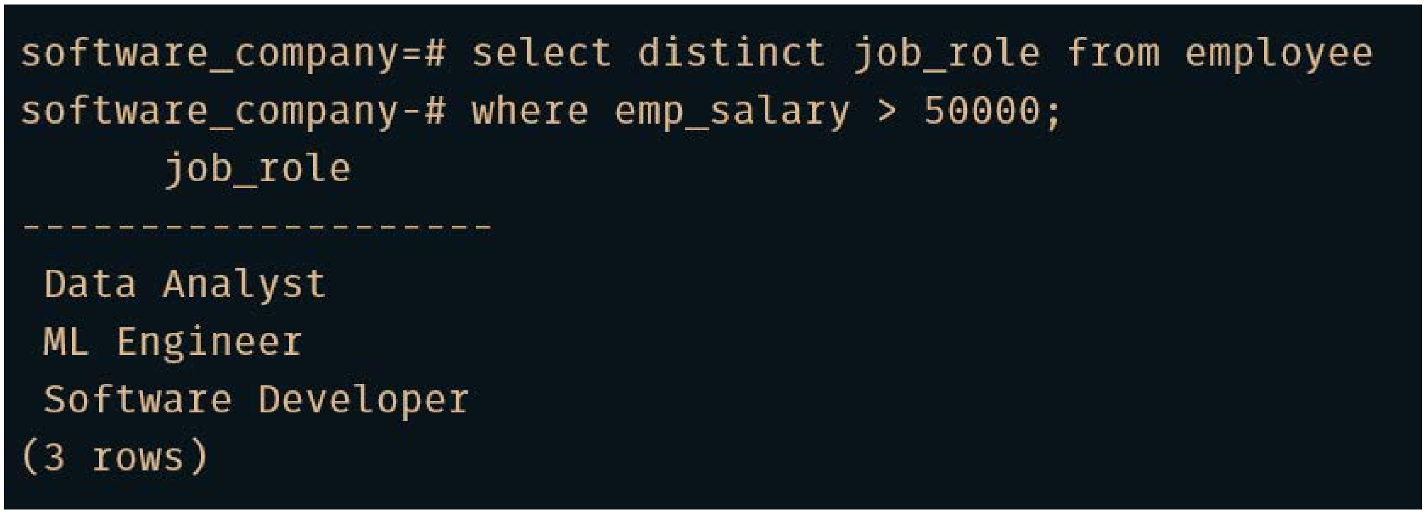
-(a) Salary is in between [50000,59999] using where clause.



```
software_company=# select distinct job_role from employee
where emp_salary BETWEEN 50000 AND 59999;
      job_role
--------------------
 Data Analyst
 ML Engineer
 Software Developer
(3 rows)
```

Retreiving distinct job roles uing select distinct job_role condition for salary b/w [50000, 59999] using the where clause condition and BETWEEN operator.

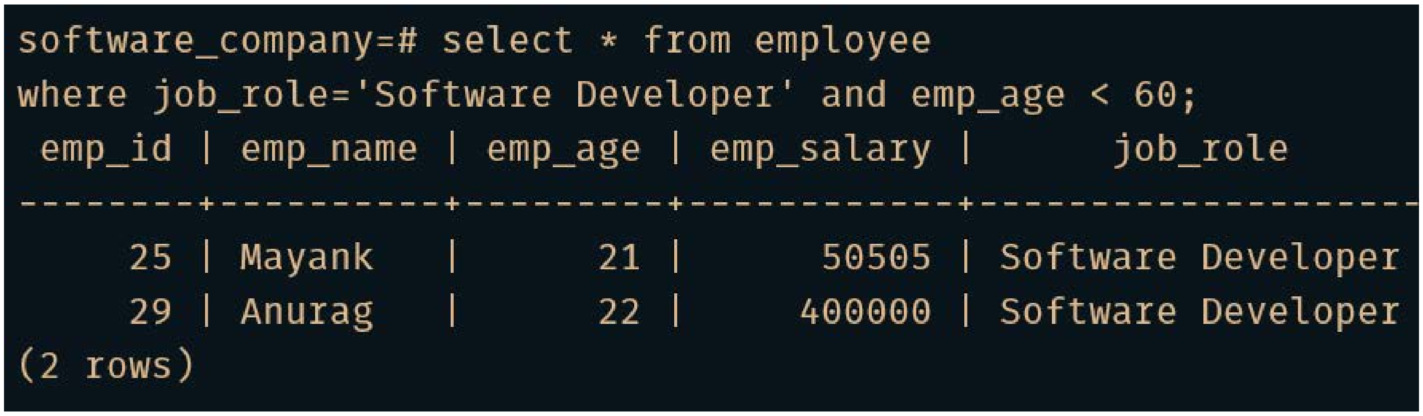
(b) Salary is greater than 50000.



```
software_company=# select distinct job_role from employee
software_company-# where emp_salary > 50000;
      job_role
--------------------
 Data Analyst
 ML Engineer
 Software Developer
(3 rows)
```

Retreiving distinct job roles uing select distinct job_role
condition for salary b/w [50000, 59999] using the where clause condition and
> operator.

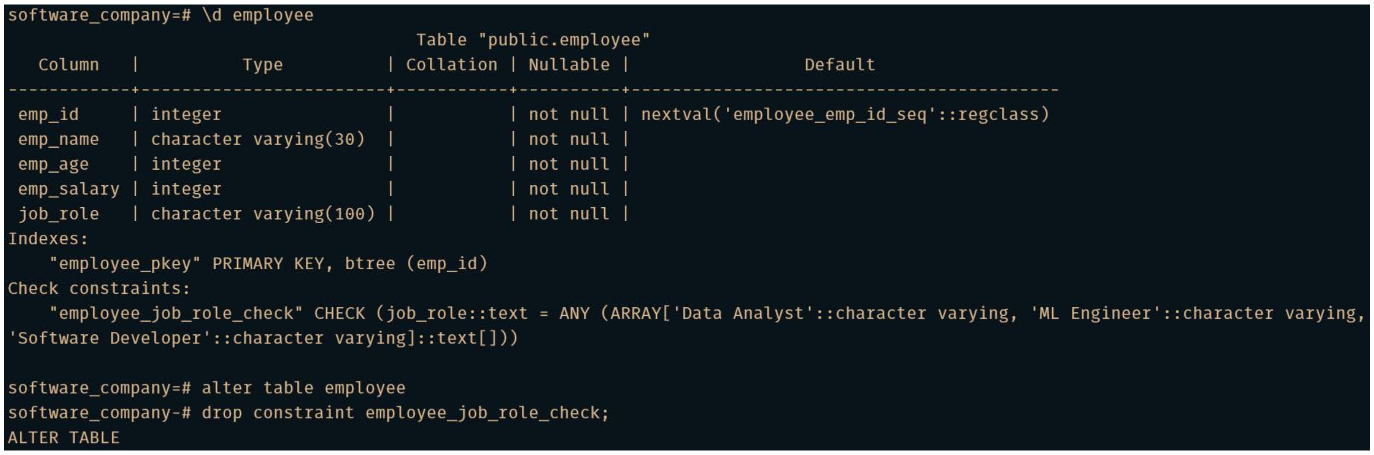
Q4. List all software developers whose age is less than 60.



```
software_company=# select * from employee
where job_role='Software Developer' and emp_age < 60;
 emp_id | emp_name | emp_age | emp_salary |      job_role
--------+----------+---------+------------+--------------------
     25 | Mayank   |      21 |      50505 | Software Developer
     29 | Anurag   |      22 |     400000 | Software Developer
(2 rows)
```

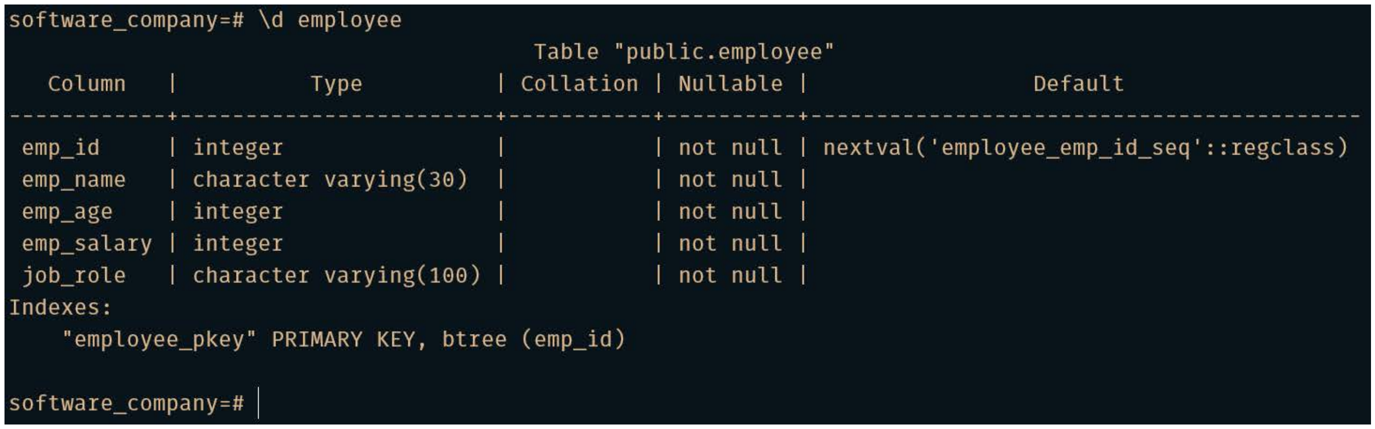
Listing all the software developers, so I select all the columns of the table using select * from employee
And put the condition in where clause that job role is Software Developer and employee age is less than 60
using the and operator.

Q5. Modify the Data Analyst job role into Data Scientist. Show the difference in output after modifying changes.



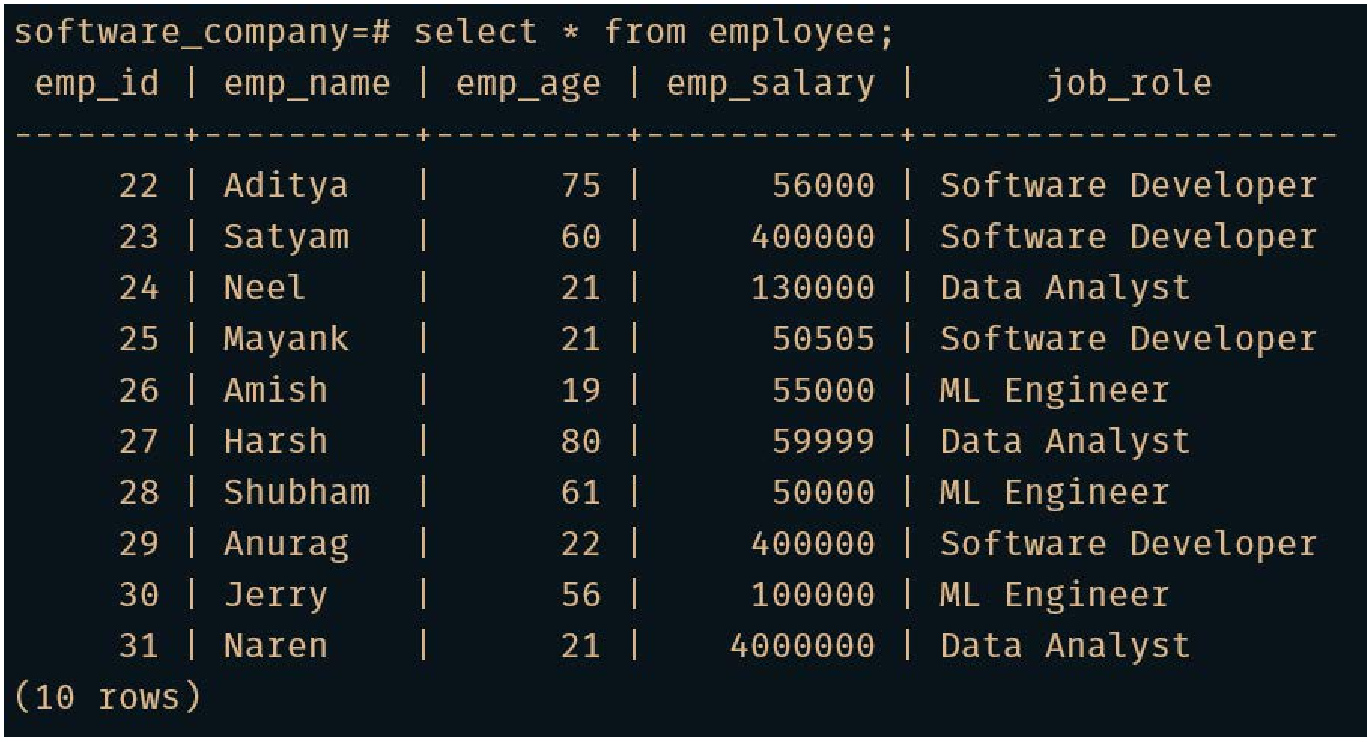
```
software_company=# \d employee
                                        Table "public.employee"
   Column   |          Type          | Collation | Nullable |                 Default
------------+------------------------+-----------+----------+------------------------------------------
 emp_id     | integer                |           | not null | nextval('employee_emp_id_seq'::regclass)
 emp_name   | character varying(30)  |           | not null |
 emp_age    | integer                |           | not null |
 emp_salary | integer                |           | not null |
 job_role   | character varying(100) |           | not null |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (emp_id)
Check constraints:
    "employee_job_role_check" CHECK (job_role::text = ANY (ARRAY['Data Analyst'::character varying, 'ML Engineer'::character varying,
'Software Developer'::character varying]::text[]))

software_company=# alter table employee
software_company-# drop constraint employee_job_role_check;
ALTER TABLE
```

We can see the constraint name that was created while creating the table by doing \d <table name>
First, we are dropping that constraint in order to change the values of the columns
using alter table <table_name> drop constraint <constraint_name>



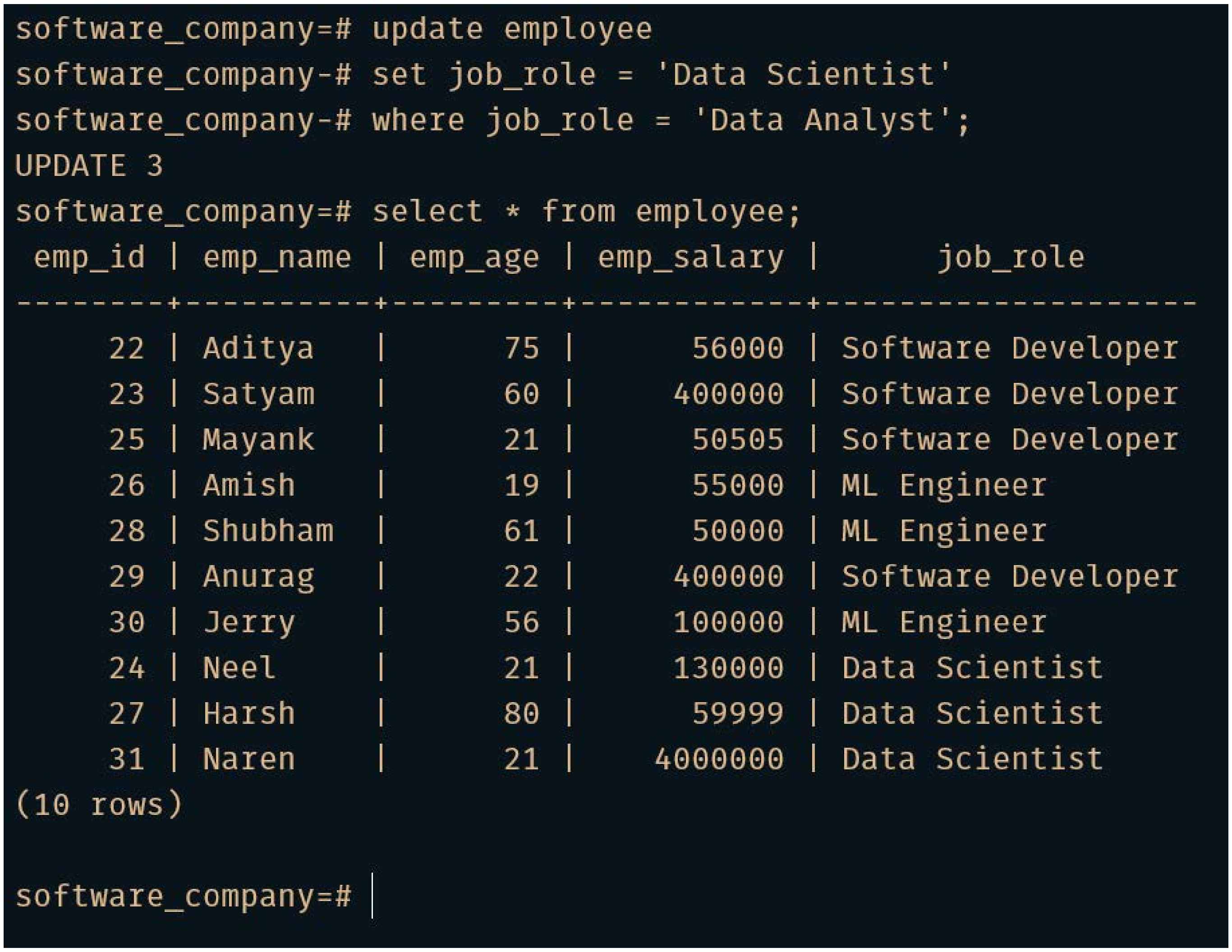
```
software_company=# \d employee
                                        Table "public.employee"
   Column   |          Type          | Collation | Nullable |                 Default
------------+------------------------+-----------+----------+------------------------------------------
 emp_id     | integer                |           | not null | nextval('employee_emp_id_seq'::regclass)
 emp_name   | character varying(30)  |           | not null |
 emp_age    | integer                |           | not null |
 emp_salary | integer                |           | not null |
 job_role   | character varying(100) |           | not null |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (emp_id)

software_company=# 
```

We can verify that the constraint is now removed.



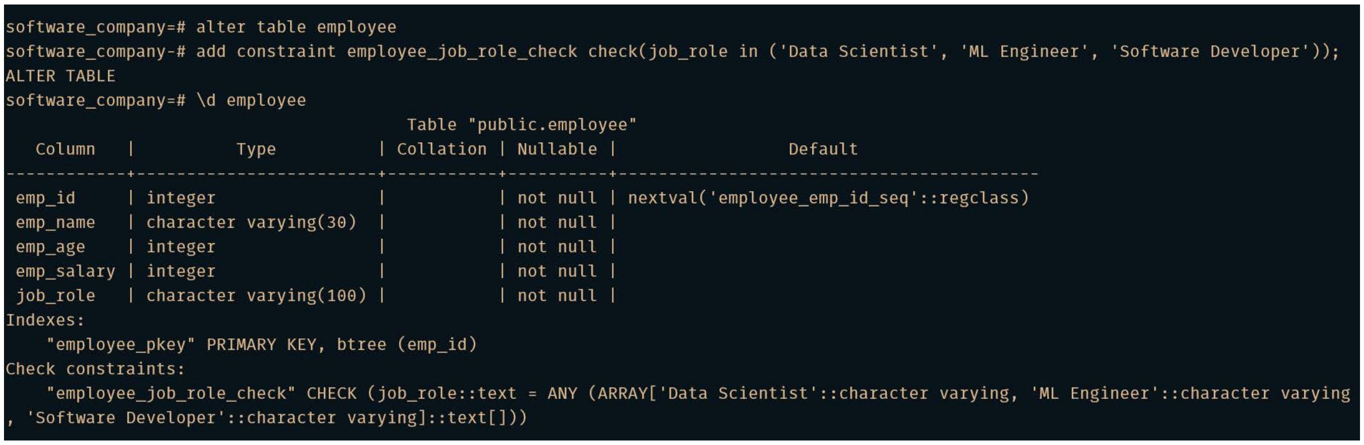
```
software_company=# select * from employee;
 emp_id | emp_name | emp_age | emp_salary |      job_role
--------+----------+---------+------------+--------------------
     22 | Aditya   |      75 |      56000 | Software Developer
     23 | Satyam   |      60 |     400000 | Software Developer
     24 | Neel     |      21 |     130000 | Data Analyst
     25 | Mayank   |      21 |      50505 | Software Developer
     26 | Amish    |      19 |      55000 | ML Engineer
     27 | Harsh    |      80 |      59999 | Data Analyst
     28 | Shubham  |      61 |      50000 | ML Engineer
     29 | Anurag   |      22 |     400000 | Software Developer
     30 | Jerry    |      56 |     100000 | ML Engineer
     31 | Naren    |      21 |    4000000 | Data Analyst
(10 rows)
```

The old table state.



```
software_company=# update employee
software_company-# set job_role = 'Data Scientist'
software_company-# where job_role = 'Data Analyst';
UPDATE 3
software_company=# select * from employee;
 emp_id | emp_name | emp_age | emp_salary |      job_role
--------+----------+---------+------------+--------------------
     22 | Aditya   |      75 |      56000 | Software Developer
     23 | Satyam   |      60 |     400000 | Software Developer
     25 | Mayank   |      21 |      50505 | Software Developer
     26 | Amish    |      19 |      55000 | ML Engineer
     28 | Shubham  |      61 |      50000 | ML Engineer
     29 | Anurag   |      22 |     400000 | Software Developer
     30 | Jerry    |      56 |     100000 | ML Engineer
     24 | Neel     |      21 |     130000 | Data Scientist
     27 | Harsh    |      80 |      59999 | Data Scientist
     31 | Naren    |      21 |    4000000 | Data Scientist
(10 rows)

software_company=#
```

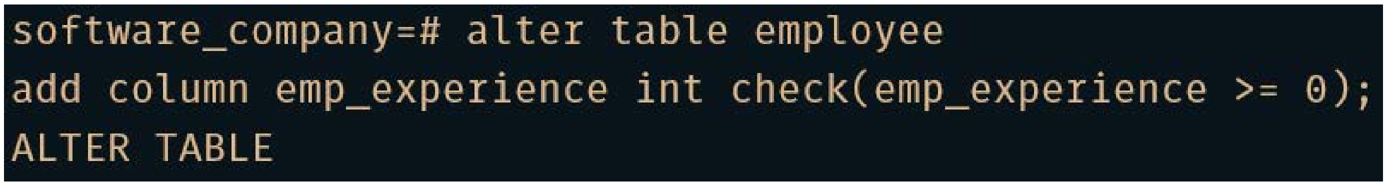
Displaying the new state of the table after updation.
Updating all the values in the column where job role was Data Anaylyst to Data Scientist.
This is done using the UPDATE <table_name> SET column=value WHERE condition



```
software_company=# alter table employee
software_company-# add constraint employee_job_role_check check(job_role in ('Data Scientist', 'ML Engineer', 'Software Developer'));
ALTER TABLE
software_company=# \d employee
                                       Table "public.employee"
   Column   |          Type          | Collation | Nullable |                Default
------------+------------------------+-----------+----------+---------------------------------------
 emp_id     | integer                |           | not null | nextval('employee_emp_id_seq'::regclass)
 emp_name   | character varying(30)  |           | not null |
 emp_age    | integer                |           | not null |
 emp_salary | integer                |           | not null |
 job_role   | character varying(100) |           | not null |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (emp_id)
Check constraints:
    "employee_job_role_check" CHECK (job_role::text = ANY (ARRAY['Data Scientist'::character varying, 'ML Engineer'::character varying
, 'Software Developer'::character varying]::text[]))
```

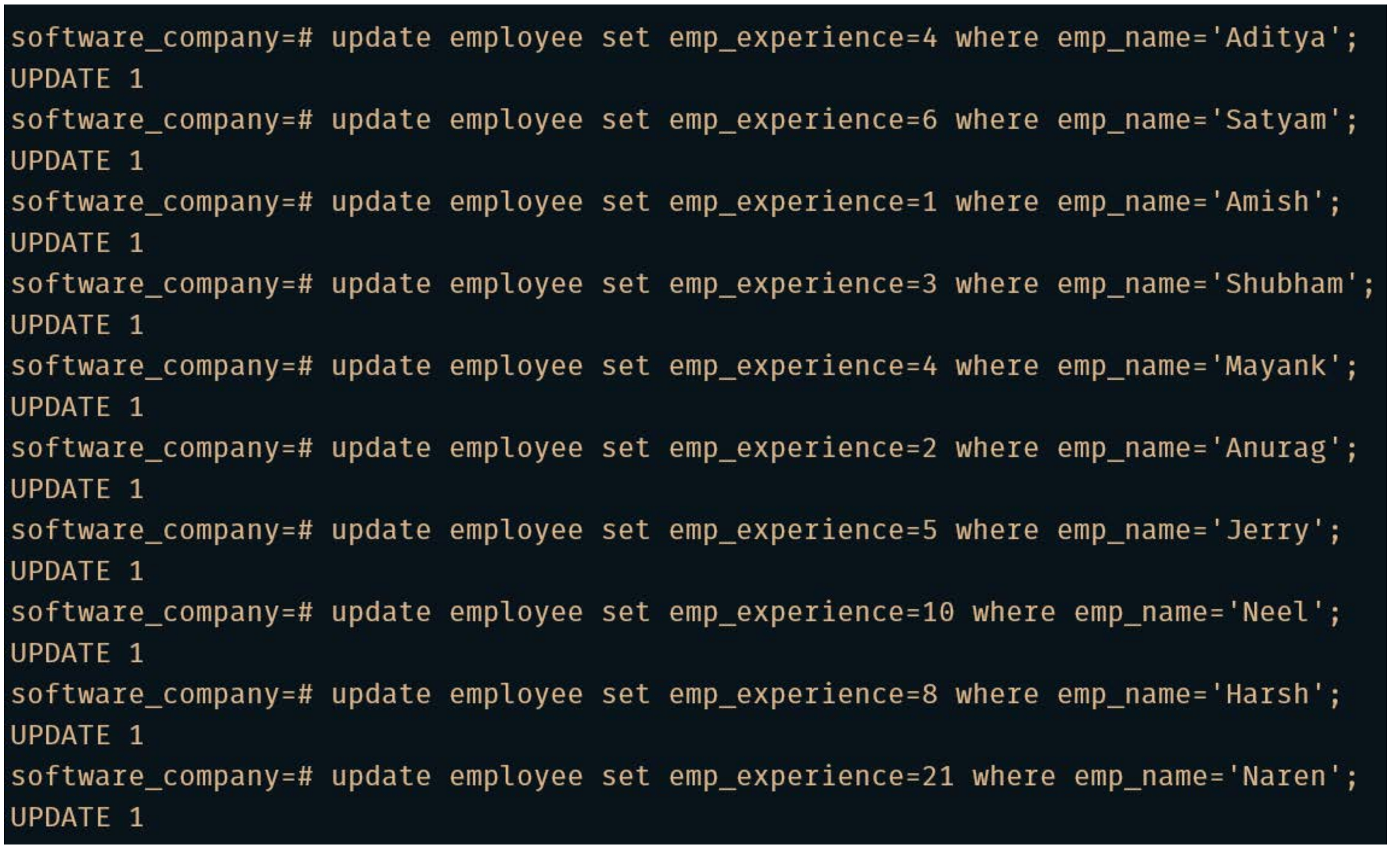
Adding constraint back to the table for the column job_role with new values using
alter table <table_name> add constraint <constraint_name> <constraint>

Q6. Add another column emp experience and insert the data into this column. Column should not contain any negative values.



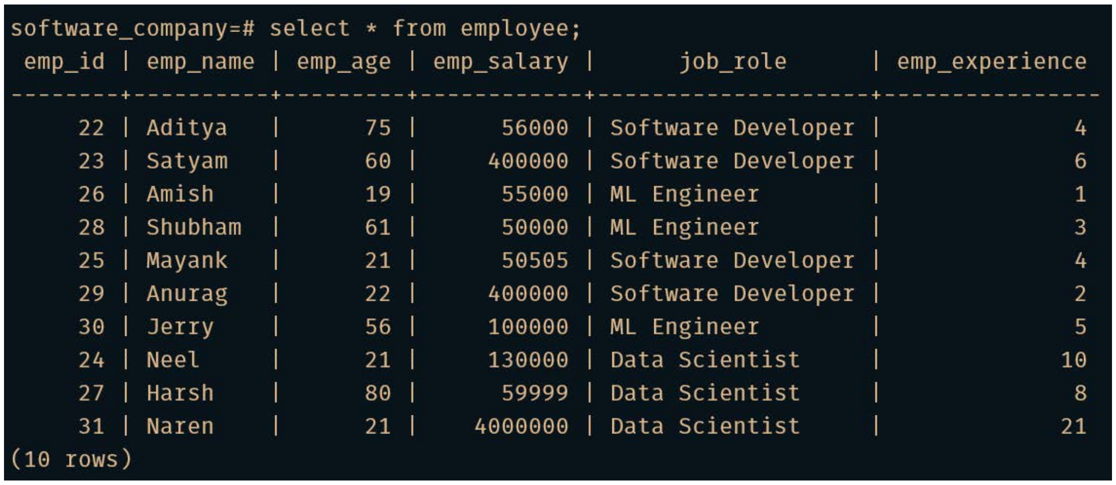
```
software_company=# alter table employee
add column emp_experience int check(emp_experience >= 0);
ALTER TABLE
```

Adding another column emp_experience to the table
with appropriate data type and check for non-negative values
using alter table <table_name> add column <column_name> <type> <constraint>



```
software_company=# update employee set emp_experience=4 where emp_name='Aditya';
UPDATE 1
software_company=# update employee set emp_experience=6 where emp_name='Satyam';
UPDATE 1
software_company=# update employee set emp_experience=1 where emp_name='Amish';
UPDATE 1
software_company=# update employee set emp_experience=3 where emp_name='Shubham';
UPDATE 1
software_company=# update employee set emp_experience=4 where emp_name='Mayank';
UPDATE 1
software_company=# update employee set emp_experience=2 where emp_name='Anurag';
UPDATE 1
software_company=# update employee set emp_experience=5 where emp_name='Jerry';
UPDATE 1
software_company=# update employee set emp_experience=10 where emp_name='Neel';
UPDATE 1
software_company=# update employee set emp_experience=8 where emp_name='Harsh';
UPDATE 1
software_company=# update employee set emp_experience=21 where emp_name='Naren';
UPDATE 1
```

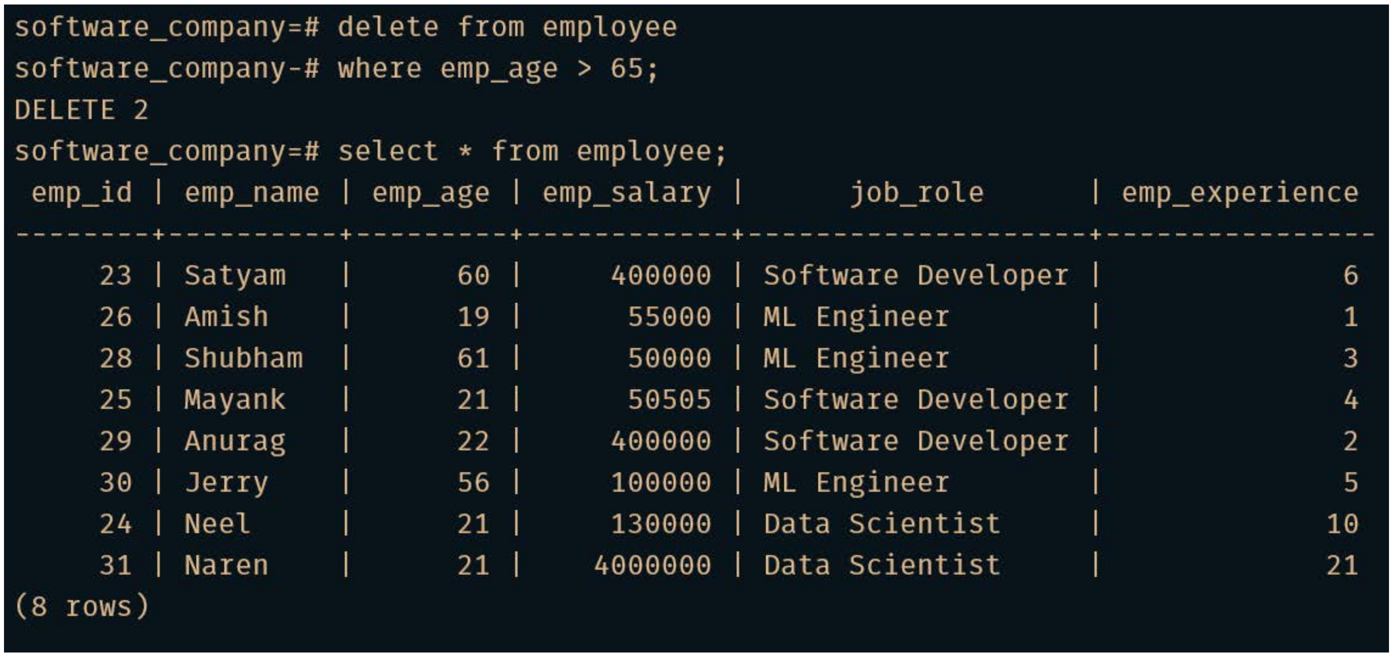
Adding data to the newly created table by updating each row separately
using update <table_name> set <column>=<value> where condition;



```
software_company=# select * from employee;
 emp_id | emp_name | emp_age | emp_salary |      job_role      | emp_experience
--------+----------+---------+------------+--------------------+----------------
     22 | Aditya   |      75 |      56000 | Software Developer |              4
     23 | Satyam   |      60 |     400000 | Software Developer |              6
     26 | Amish    |      19 |      55000 | ML Engineer        |              1
     28 | Shubham  |      61 |      50000 | ML Engineer        |              3
     25 | Mayank   |      21 |      50505 | Software Developer |              4
     29 | Anurag   |      22 |     400000 | Software Developer |              2
     30 | Jerry    |      56 |     100000 | ML Engineer        |              5
     24 | Neel     |      21 |     130000 | Data Scientist     |             10
     27 | Harsh    |      80 |      59999 | Data Scientist     |              8
     31 | Naren    |      21 |    4000000 | Data Scientist     |             21
(10 rows)
```

Displaying the final state of the table.

Q7. Delete all employees whose age is greater than 65.



```
software_company=# delete from employee
software_company-# where emp_age > 65;
DELETE 2
software_company=# select * from employee;
 emp_id | emp_name | emp_age | emp_salary |      job_role       | emp_experience
--------+----------+---------+------------+---------------------+----------------
     23 | Satyam   |      60 |     400000 | Software Developer |              6
     26 | Amish    |      19 |      55000 | ML Engineer         |              1
     28 | Shubham  |      61 |      50000 | ML Engineer         |              3
     25 | Mayank   |      21 |      50505 | Software Developer |              4
     29 | Anurag   |      22 |     400000 | Software Developer |              2
     30 | Jerry    |      56 |     100000 | ML Engineer         |              5
     24 | Neel     |      21 |     130000 | Data Scientist      |             10
     31 | Naren    |      21 |    4000000 | Data Scientist      |             21
(8 rows)
```

Displaying the final state of the table after deletion.
Deleting the employees with age > 65
using delete from <table_name> where <condition>