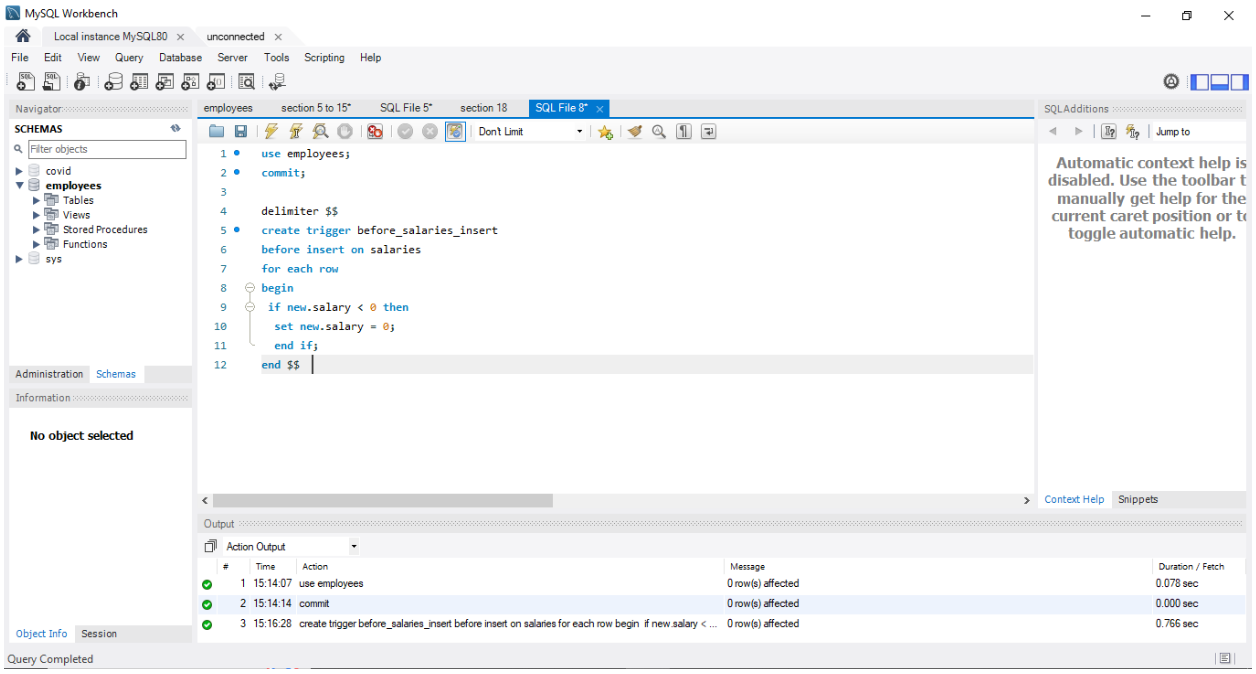
SECTION 19

Triggers:

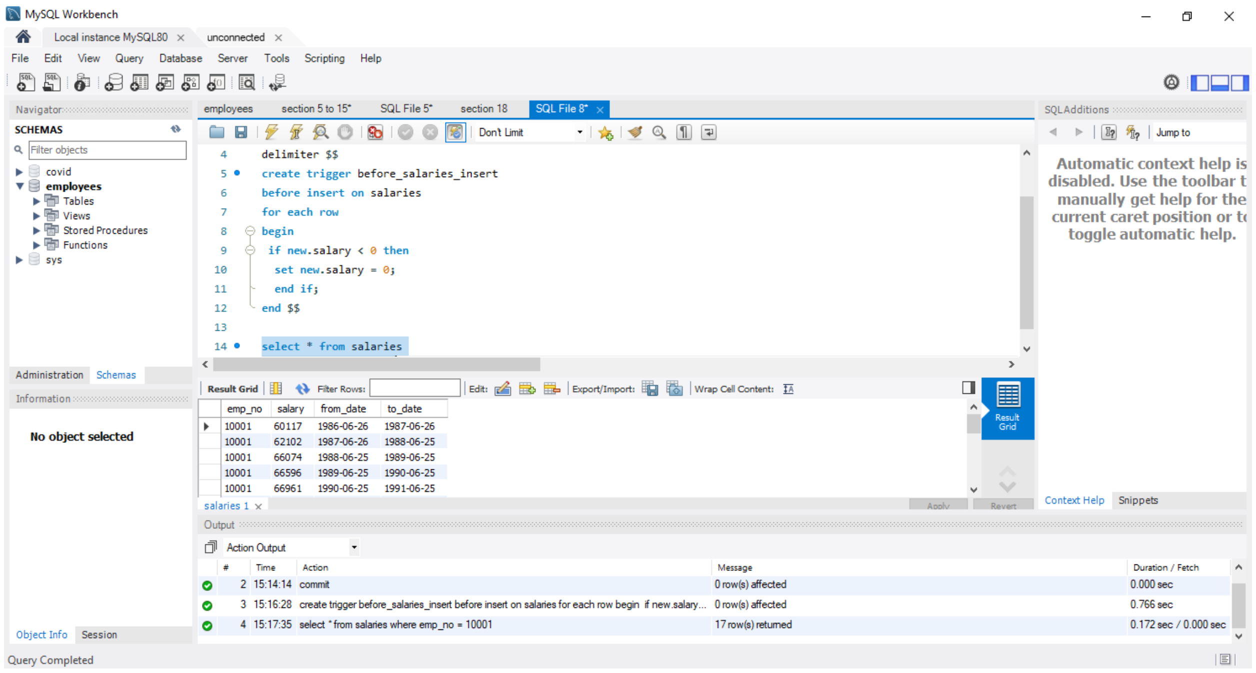
We got introduced to a new concept; triggers. Following is the example that demonstrates how trigger works:



```
use employees;
commit;

delimiter $$
create trigger before_salaries_insert
before insert on salaries
for each row
begin
 if new.salary < 0 then
  set new.salary = 0;
  end if;
end $$
```

Output — Action Output

```
#  Time      Action                                                                          Message            Duration / Fetch
1  15:14:07  use employees                                                                   0 row(s) affected  0.078 sec
2  15:14:14  commit                                                                          0 row(s) affected  0.000 sec
3  15:16:28  create trigger before_salaries_insert before insert on salaries for each row begin  if new.salary < ...  0 row(s) affected  0.766 sec
```

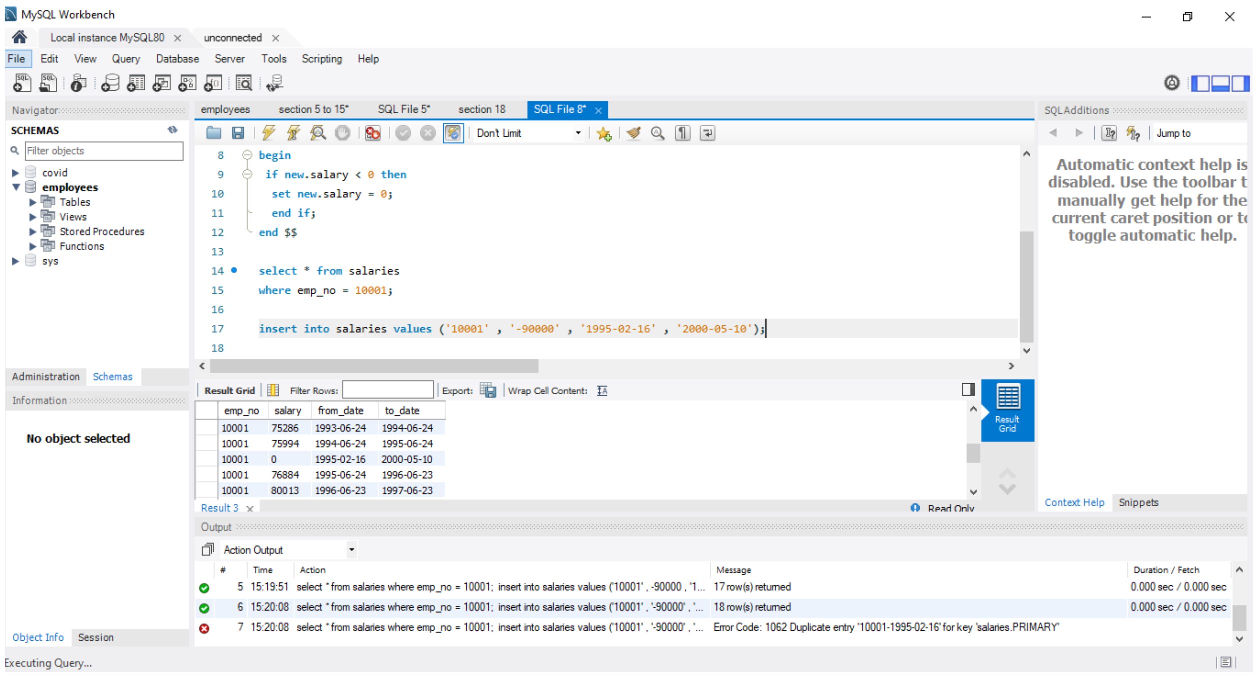


```
delimiter $$
create trigger before_salaries_insert
before insert on salaries
for each row
begin
 if new.salary < 0 then
  set new.salary = 0;
  end if;
end $$

select * from salaries
```

```
emp_no  salary  from_date   to_date
10001   60117   1986-06-26  1987-06-26
10001   62102   1987-06-26  1988-06-25
10001   66074   1988-06-25  1989-06-25
10001   66596   1989-06-25  1990-06-25
10001   66961   1990-06-25  1991-06-25
```

Output — Action Output

```
#  Time      Action                                                                          Message            Duration / Fetch
2  15:14:14  commit                                                                          0 row(s) affected  0.000 sec
3  15:16:28  create trigger before_salaries_insert before insert on salaries for each row begin  if new.salary...  0 row(s) affected  0.766 sec
4  15:17:35  select * from salaries where emp_no = 10001                                     17 row(s) returned  0.172 sec / 0.000 sec
```



```
begin
 if new.salary < 0 then
  set new.salary = 0;
  end if;
end $$

select * from salaries
where emp_no = 10001;

insert into salaries values ('10001' , '-90000' , '1995-02-16' , '2000-05-10');
```

emp_no	salary	from_date	to_date
10001	75286	1993-06-24	1994-06-24
10001	75994	1994-06-24	1995-06-24
10001	0	1995-02-16	2000-05-10
10001	76884	1995-06-24	1996-06-23
10001	80013	1996-06-23	1997-06-23

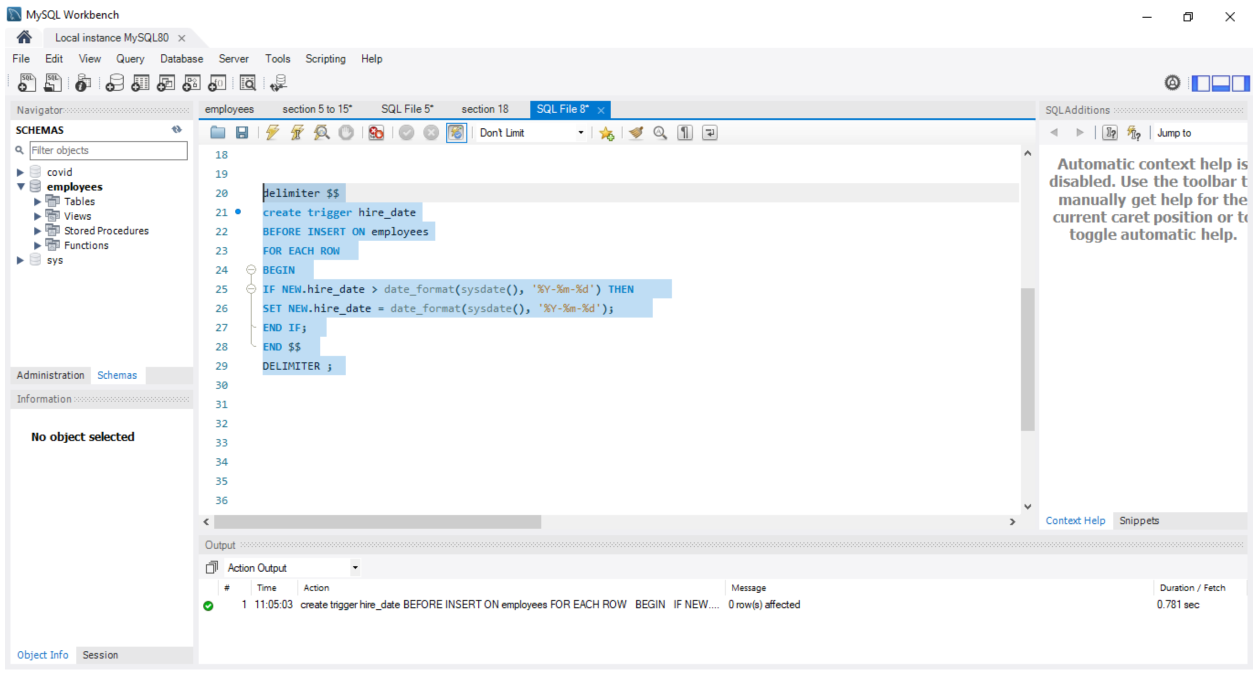
```
#  Time      Action                                                                           Message                                                                      Duration / Fetch
5  15:19:51  select * from salaries where emp_no = 10001;  insert into salaries values ('10001' , -90000 , '1...   17 row(s) returned                                                           0.000 sec / 0.000 sec
6  15:20:08  select * from salaries where emp_no = 10001;  insert into salaries values ('10001' , '-90000' , '...   18 row(s) returned                                                           0.000 sec / 0.000 sec
7  15:20:08  select * from salaries where emp_no = 10001;  insert into salaries values ('10001' , '-90000' , '...   Error Code: 1062 Duplicate entry '10001-1995-02-16' for key 'salaries.PRIMARY'
```

Executing Query...

Exercise:

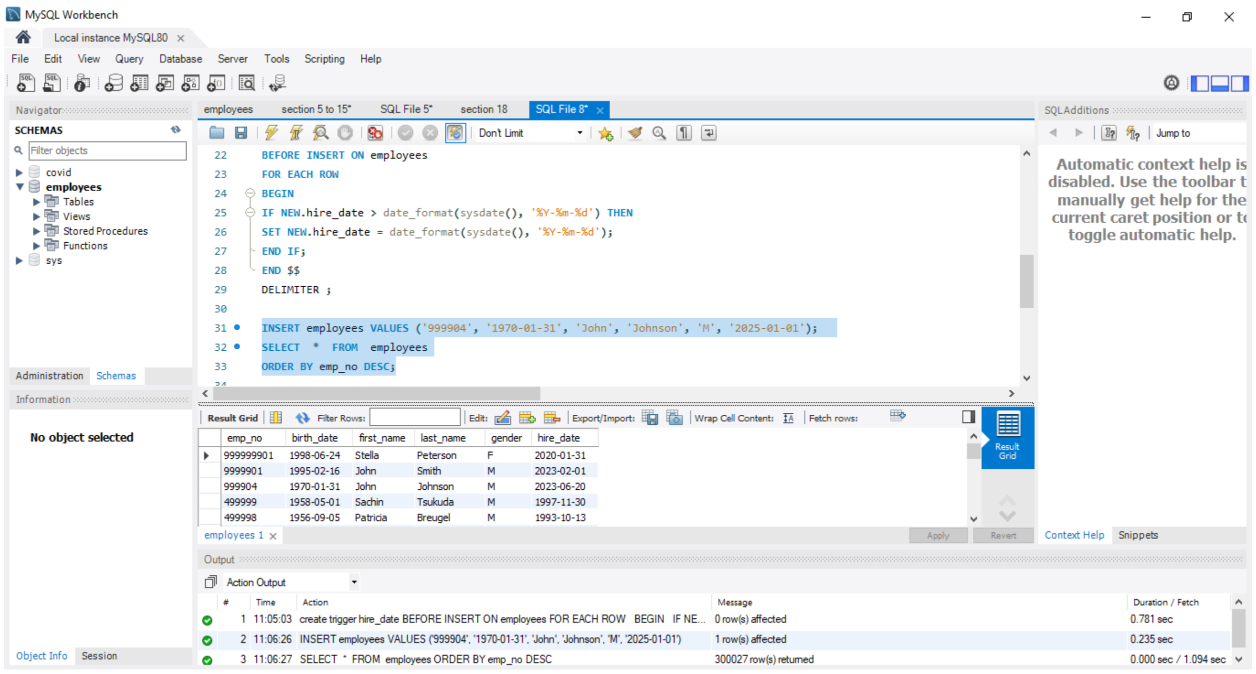
Create a trigger that checks if the hire date of an employee is higher than the current date. If true, set this date to be the current date. Format the output appropriately (YY-MM-DD).

Solution:



```
delimiter $$
create trigger hire_date
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
IF NEW.hire_date > date_format(sysdate(), '%Y-%m-%d') THEN
SET NEW.hire_date = date_format(sysdate(), '%Y-%m-%d');
END IF;
END $$
DELIMITER ;
```

```
#  Time      Action                                                                    Message              Duration / Fetch
1  11:05:03  create trigger hire_date BEFORE INSERT ON employees FOR EACH ROW   BEGIN   IF NEW....   0 row(s) affected    0.781 sec
```



```sql
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
IF NEW.hire_date > date_format(sysdate(), '%Y-%m-%d') THEN
SET NEW.hire_date = date_format(sysdate(), '%Y-%m-%d');
END IF;
END $$
DELIMITER ;

INSERT employees VALUES ('999904', '1970-01-31', 'John', 'Johnson', 'M', '2025-01-01');
SELECT  *  FROM  employees
ORDER BY emp_no DESC;
```

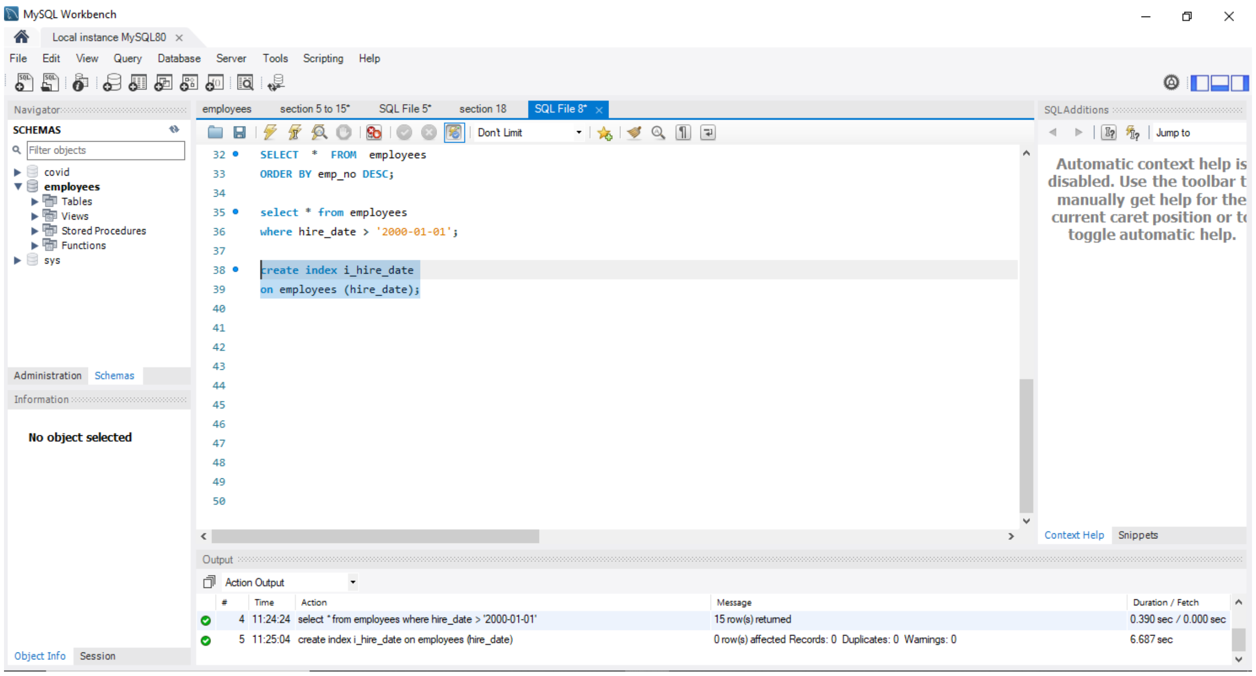
emp_no	birth_date	first_name	last_name	gender	hire_date
999999901	1998-06-24	Stella	Peterson	F	2020-01-31
9999901	1995-02-16	John	Smith	M	2023-02-01
999904	1970-01-31	John	Johnson	M	2023-06-20
499999	1958-05-01	Sachin	Tsukuda	M	1997-11-30
499998	1956-09-05	Patricia	Breugel	M	1993-10-13

```
#  Time      Action                                                                         Message                 Duration / Fetch
1  11:05:03  create trigger hire_date BEFORE INSERT ON employees FOR EACH ROW  BEGIN  IF NE...  0 row(s) affected       0.781 sec
2  11:06:26  INSERT employees VALUES ('999904', '1970-01-31', 'John', 'Johnson', 'M', '2025-01-01')  1 row(s) affected  0.235 sec
3  11:06:27  SELECT  *  FROM  employees ORDER BY emp_no DESC                                   300027 row(s) returned  0.000 sec / 1.094 sec
```

<u>Indexes:</u>

Now, imagine you want to frequently sort the peoples in the employees table according to their hire dates. You must use indexes.

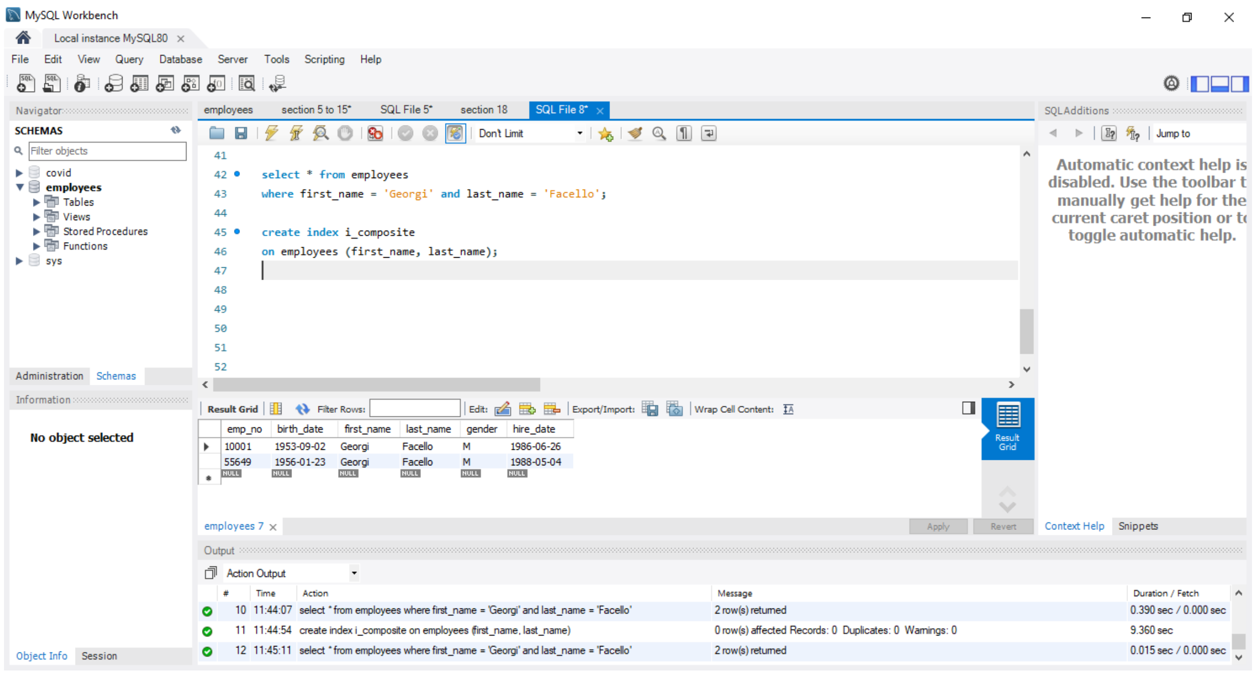
Run a query that tells us how many people were hired after 1st January '2000.



```sql
SELECT  *  FROM  employees
ORDER BY emp_no DESC;

select * from employees
where hire_date > '2000-01-01';

create index i_hire_date
on employees (hire_date);
```

```
#  Time      Action                                                    Message                                                    Duration / Fetch
4  11:24:24  select * from employees where hire_date > '2000-01-01'   15 row(s) returned                                         0.390 sec / 0.000 sec
5  11:25:04  create index i_hire_date on employees (hire_date)        0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   6.687 sec
```

Select all employees bearing the name Georgi Facello using composite indexes



```sql
select * from employees
where first_name = 'Georgi' and last_name = 'Facello';

create index i_composite
on employees (first_name, last_name);
```

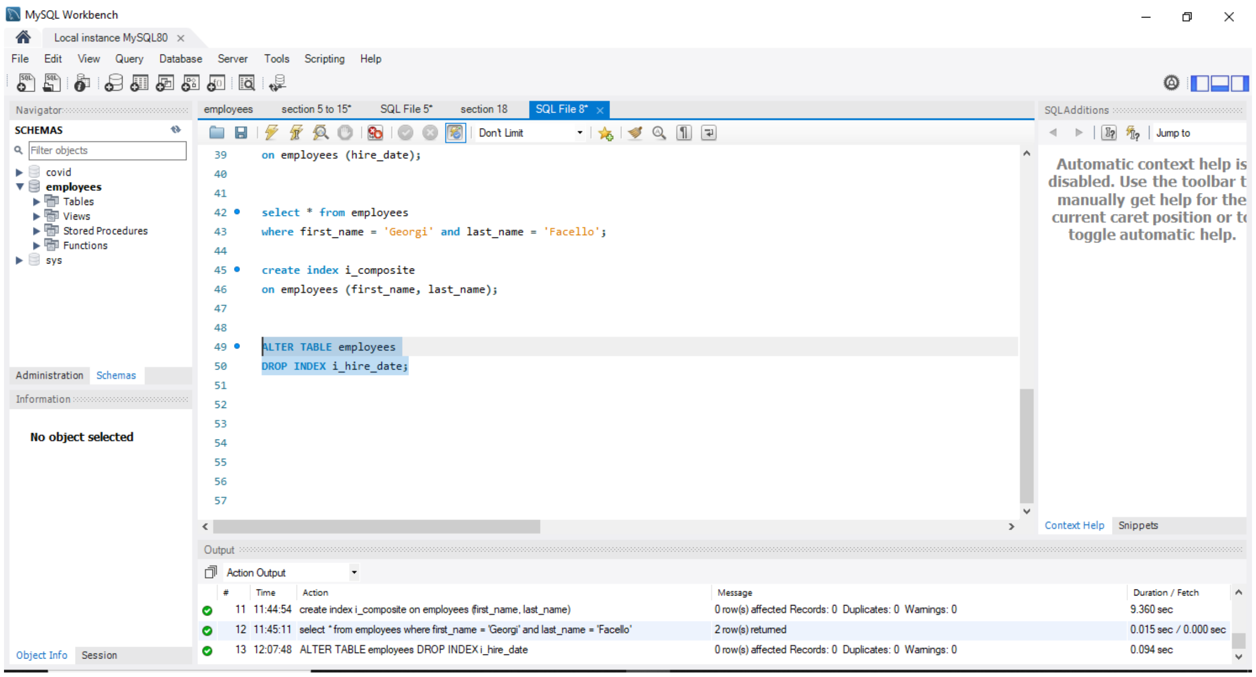
emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
55649	1956-01-23	Georgi	Facello	M	1988-05-04
NULL	NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                 Message                                                  Duration / Fetch
10  11:44:07  select * from employees where first_name = 'Georgi' and last_name = 'Facello'   2 row(s) returned                                        0.390 sec / 0.000 sec
11  11:44:54  create index i_composite on employees (first_name, last_name)          0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   9.360 sec
12  11:45:11  select * from employees where first_name = 'Georgi' and last_name = 'Facello'   2 row(s) returned                                        0.015 sec / 0.000 sec
```

Exercise 1:

Drop the ‘i_hire_date’ index.

Solution 1:



```sql
on employees (hire_date);


select * from employees
where first_name = 'Georgi' and last_name = 'Facello';

create index i_composite
on employees (first_name, last_name);


ALTER TABLE employees
DROP INDEX i_hire_date;
```

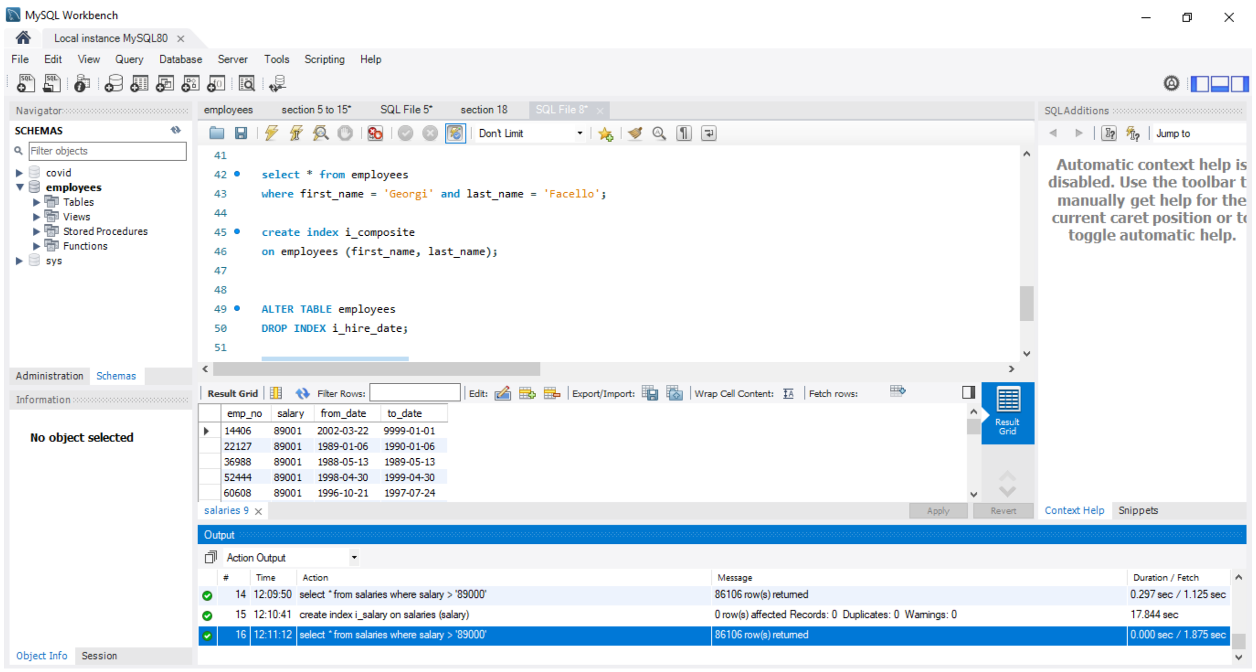
```
#   Time      Action                                                                 Message                                                  Duration / Fetch
11  11:44:54  create index i_composite on employees (first_name, last_name)          0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   9.360 sec
12  11:45:11  select * from employees where first_name = 'Georgi' and last_name = 'Facello'   2 row(s) returned                                        0.015 sec / 0.000 sec
13  12:07:48  ALTER TABLE employees DROP INDEX i_hire_date                            0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.094 sec
```

Exercise 2:

Select all records from the ‘salaries’ table of people whose salary is higher than $89,000 per annum.

Then, create an index on the 'salary' column of that table, and check if it has sped up the search of the same SELECT statement.

Solution 2:



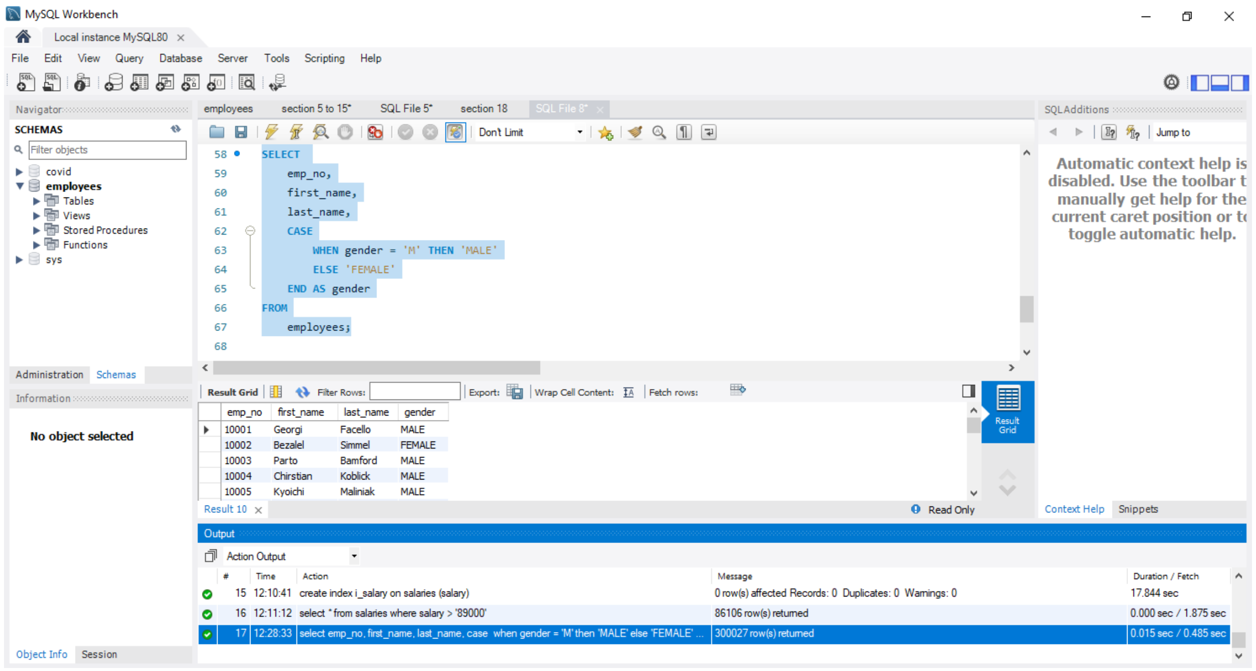
```sql
select * from employees
where first_name = 'Georgi' and last_name = 'Facello';

create index i_composite
on employees (first_name, last_name);


ALTER TABLE employees
DROP INDEX i_hire_date;
```

emp_no	salary	from_date	to_date
14406	89001	2002-03-22	9999-01-01
22127	89001	1989-01-06	1990-01-06
36988	89001	1988-05-13	1989-05-13
52444	89001	1998-04-30	1999-04-30
60608	89001	1996-10-21	1997-07-24

#	Time	Action	Message	Duration / Fetch
14	12:09:50	select * from salaries where salary > '89000'	86106 row(s) returned	0.297 sec / 1.125 sec
15	12:10:41	create index i_salary on salaries (salary)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	17.844 sec
16	12:11:12	select * from salaries where salary > '89000'	86106 row(s) returned	0.000 sec / 1.875 sec

Yes, the speed was increase.

The CASE statement:



```sql
SELECT
    emp_no,
    first_name,
    last_name,
    CASE
        WHEN gender = 'M' THEN 'MALE'
        ELSE 'FEMALE'
    END AS gender
FROM
    employees;
```

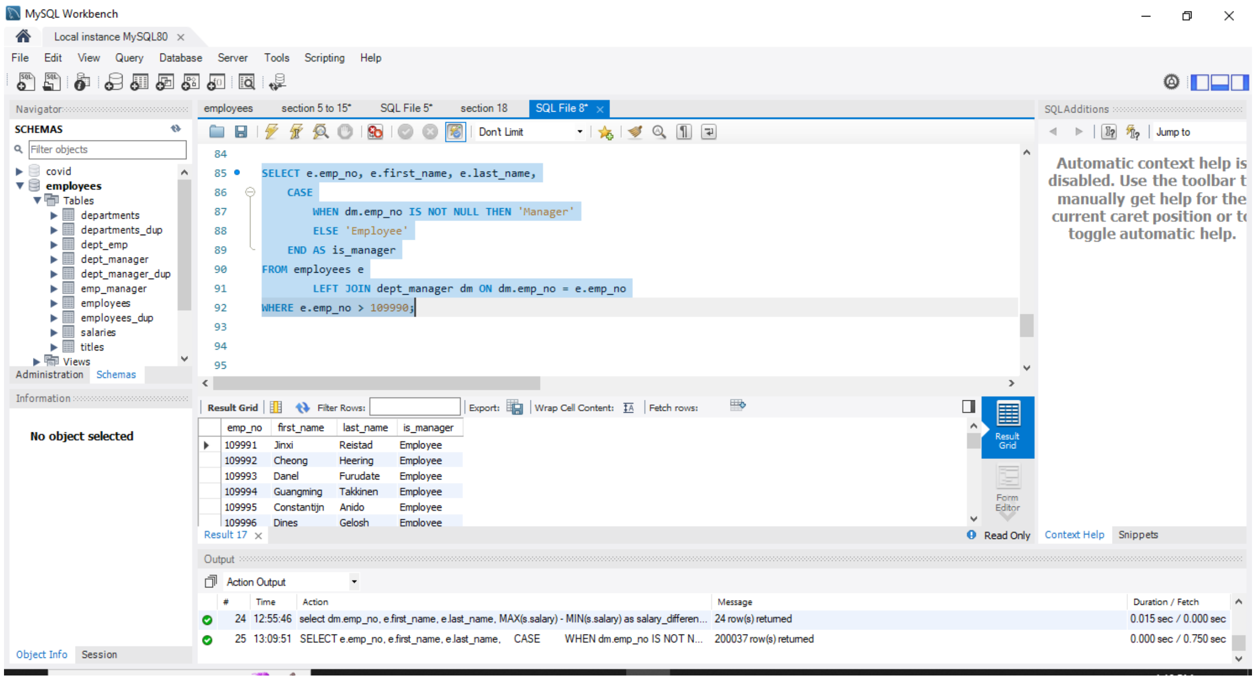
emp_no	first_name	last_name	gender
10001	Georgi	Facello	MALE
10002	Bezalel	Simmel	FEMALE
10003	Parto	Bamford	MALE
10004	Chirstian	Koblick	MALE
10005	Kyoichi	Maliniak	MALE

#	Time	Action	Message	Duration / Fetch
15	12:10:41	create index i_salary on salaries (salary)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	17.844 sec
16	12:11:12	select * from salaries where salary > '89000'	86106 row(s) returned	0.000 sec / 1.875 sec
17	12:28:33	select emp_no, first_name, last_name, case when gender = 'M' then 'MALE' else 'FEMALE' ...	300027 row(s) returned	0.015 sec / 0.485 sec

Exercise 1:

Similar to the exercises done in the lecture, obtain a result set containing the employee number, first name, and last name of all employees with a number higher than 109990. Create a fourth column in the query, indicating whether this employee is also a manager, according to the data provided in the dept_manager table, or a regular employee.

Solution 1:



```sql
SELECT e.emp_no, e.first_name, e.last_name,
    CASE
        WHEN dm.emp_no IS NOT NULL THEN 'Manager'
        ELSE 'Employee'
    END AS is_manager
FROM employees e
        LEFT JOIN dept_manager dm ON dm.emp_no = e.emp_no
WHERE e.emp_no > 109990;
```

emp_no	first_name	last_name	is_manager
109991	Jinxi	Reistad	Employee
109992	Cheong	Heering	Employee
109993	Danel	Furudate	Employee
109994	Guangming	Takkinen	Employee
109995	Constantijn	Anido	Employee
109996	Dines	Gelosh	Employee

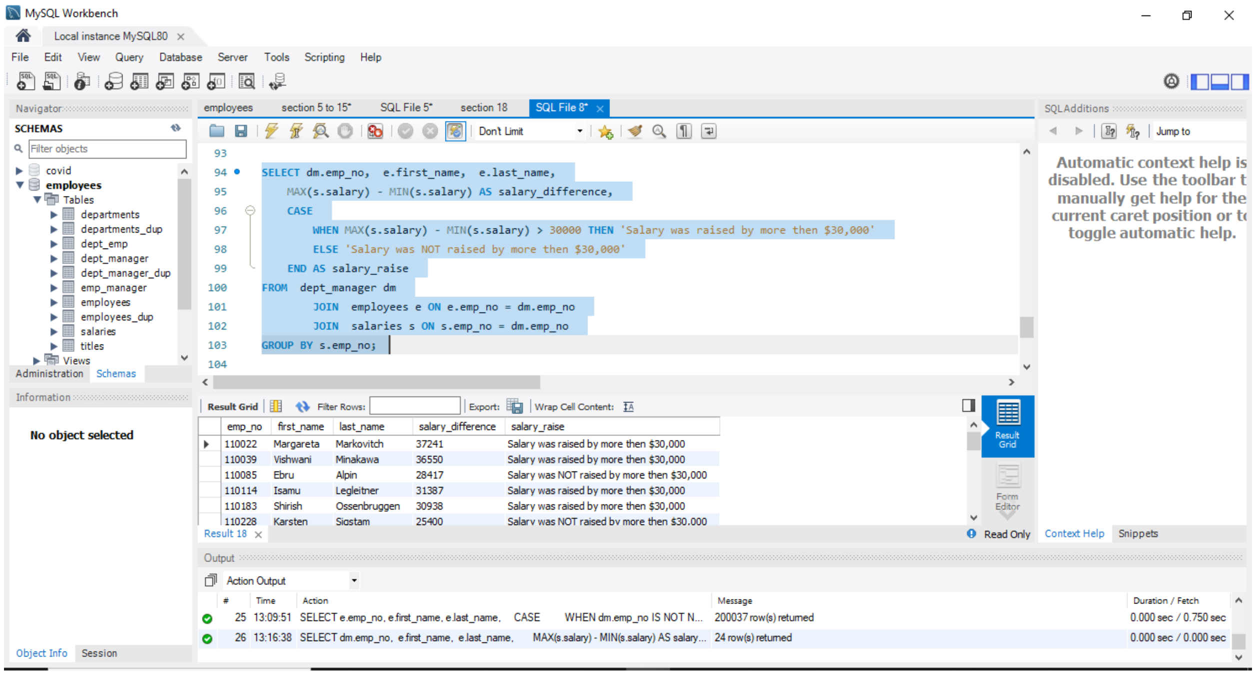
```
#   Time      Action                                                                                  Message                Duration / Fetch
24  12:55:46  select dm.emp_no, e.first_name, e.last_name, MAX(s.salary) - MIN(s.salary) as salary_differen...  24 row(s) returned     0.015 sec / 0.000 sec
25  13:09:51  SELECT e.emp_no, e.first_name, e.last_name,    CASE      WHEN dm.emp_no IS NOT N...   200037 row(s) returned   0.000 sec / 0.750 sec
```

Exercise 2:

Extract a dataset containing the following information about the managers: employee number, first name, and last name. Add two columns at the end – one showing the difference between the maximum and minimum salary of that employee, and another one saying whether this salary raise was higher than $30,000 or NOT.

If possible, provide more than one solution.

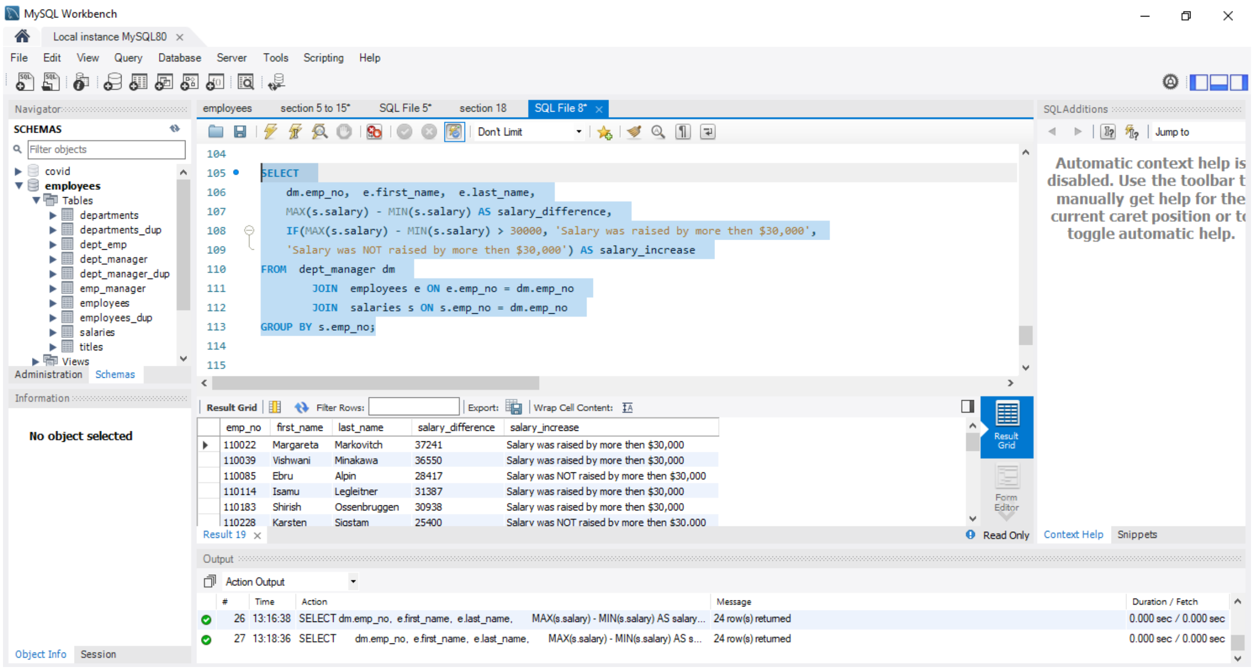
Solution 2:



```sql
SELECT dm.emp_no,  e.first_name,  e.last_name,
    MAX(s.salary) - MIN(s.salary) AS salary_difference,
    CASE
        WHEN MAX(s.salary) - MIN(s.salary) > 30000 THEN 'Salary was raised by more then $30,000'
        ELSE 'Salary was NOT raised by more then $30,000'
    END AS salary_raise
FROM  dept_manager dm
        JOIN  employees e ON e.emp_no = dm.emp_no
        JOIN  salaries s ON s.emp_no = dm.emp_no
GROUP BY s.emp_no;
```

	emp_no	first_name	last_name	salary_difference	salary_raise
▶	110022	Margareta	Markovitch	37241	Salary was raised by more then $30,000
	110039	Vishwani	Minakawa	36550	Salary was raised by more then $30,000
	110085	Ebru	Alpin	28417	Salary was NOT raised by more then $30,000
	110114	Isamu	Legleitner	31387	Salary was raised by more then $30,000
	110183	Shirish	Ossenbruggen	30938	Salary was raised by more then $30,000
	110228	Karsten	Sigstam	25400	Salary was NOT raised by more then $30,000

```
#   Time      Action                                                                          Message                   Duration / Fetch
25  13:09:51  SELECT e.emp_no, e.first_name, e.last_name,   CASE      WHEN dm.emp_no IS NOT N...  200037 row(s) returned    0.000 sec / 0.750 sec
26  13:16:38  SELECT dm.emp_no,  e.first_name,  e.last_name,    MAX(s.salary) - MIN(s.salary) AS salary...  24 row(s) returned    0.000 sec / 0.000 sec
```



```sql
SELECT
    dm.emp_no,  e.first_name,  e.last_name,
    MAX(s.salary) - MIN(s.salary) AS salary_difference,
    IF(MAX(s.salary) - MIN(s.salary) > 30000, 'Salary was raised by more then $30,000',
    'Salary was NOT raised by more then $30,000') AS salary_increase
FROM  dept_manager dm
        JOIN  employees e ON e.emp_no = dm.emp_no
        JOIN  salaries s ON s.emp_no = dm.emp_no
GROUP BY s.emp_no;
```

	emp_no	first_name	last_name	salary_difference	salary_increase
▶	110022	Margareta	Markovitch	37241	Salary was raised by more then $30,000
	110039	Vishwani	Minakawa	36550	Salary was raised by more then $30,000
	110085	Ebru	Alpin	28417	Salary was NOT raised by more then $30,000
	110114	Isamu	Legleitner	31387	Salary was raised by more then $30,000
	110183	Shirish	Ossenbruggen	30938	Salary was raised by more then $30,000
	110228	Karsten	Sigstam	25400	Salary was NOT raised by more then $30,000

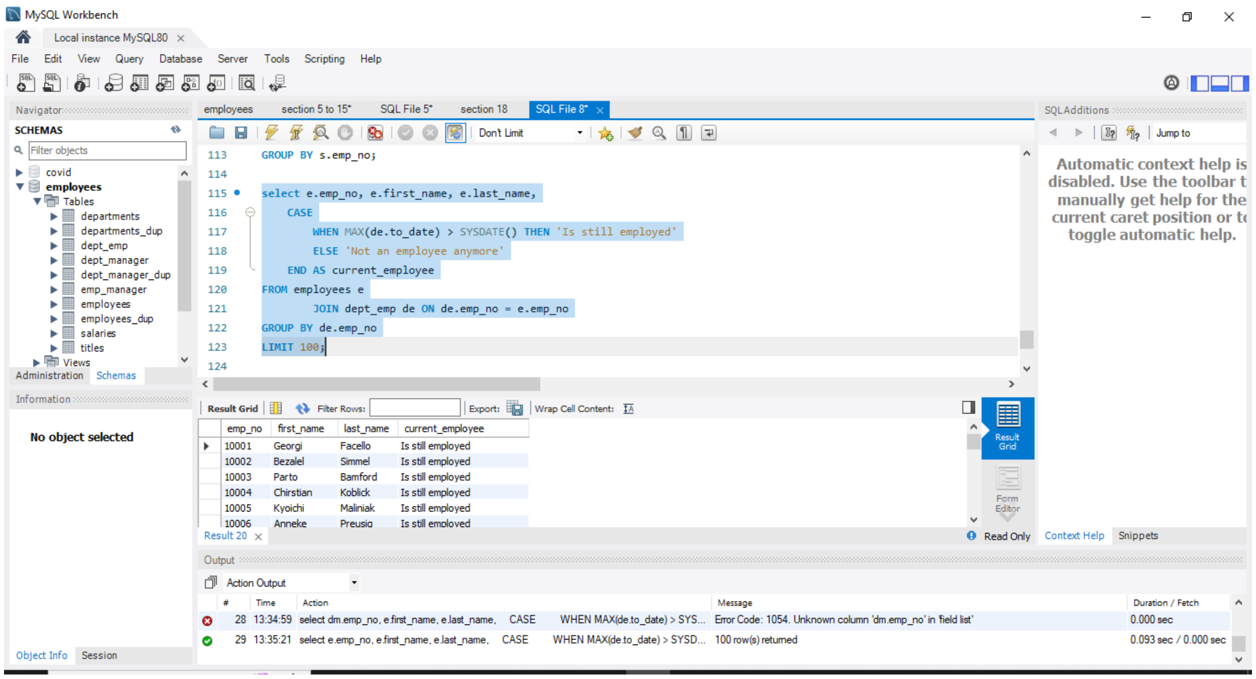
```
#   Time      Action                                                                          Message                 Duration / Fetch
26  13:16:38  SELECT dm.emp_no,  e.first_name,  e.last_name,    MAX(s.salary) - MIN(s.salary) AS salary...  24 row(s) returned    0.000 sec / 0.000 sec
27  13:18:36  SELECT     dm.emp_no,  e.first_name,  e.last_name,    MAX(s.salary) - MIN(s.salary) AS s...  24 row(s) returned    0.000 sec / 0.000 sec
```

Exercise 3:

Extract the employee number, first name, and last name of the first 100 employees, and add a fourth column, called “current_employee” saying “Is still employed” if the employee is still working in the company, or “Not an employee anymore” if they aren’t.

Hint: You’ll need to use data from both the ‘employees’ and the ‘dept_emp’ table to solve this exercise.

Solution 3:



```sql
GROUP BY s.emp_no;

select e.emp_no, e.first_name, e.last_name,
    CASE
        WHEN MAX(de.to_date) > SYSDATE() THEN 'Is still employed'
        ELSE 'Not an employee anymore'
    END AS current_employee
FROM employees e
        JOIN dept_emp de ON de.emp_no = e.emp_no
GROUP BY de.emp_no
LIMIT 100;
```

emp_no	first_name	last_name	current_employee
10001	Georgi	Facello	Is still employed
10002	Bezalel	Simmel	Is still employed
10003	Parto	Bamford	Is still employed
10004	Chirstian	Koblick	Is still employed
10005	Kyoichi	Maliniak	Is still employed
10006	Anneke	Preusig	Is still employed

```
#   Time      Action                                                                        Message                                                  Duration / Fetch
28  13:34:59  select dm.emp_no, e.first_name, e.last_name,   CASE   WHEN MAX(de.to_date) > SYS...  Error Code: 1054. Unknown column 'dm.emp_no' in 'field list'  0.000 sec
29  13:35:21  select e.emp_no, e.first_name, e.last_name,   CASE   WHEN MAX(de.to_date) > SYSD...  100 row(s) returned                                      0.093 sec / 0.000 sec
```