ADS – ITS62004 (AUG 2018) Practical Test 2 (15%) Date: 15-11-2018

You need to download and unzip a file with name Practical Test2 – scheme.zip available at Times > Week 11 and complete all Tasks below. After testing each piece of required queries, you need to save the query and a screenshot of the result of the query (if there is any) and save all the answer into a Microsoft Word Document and name the file using pattern "ITS62004-PT2-[your full name]-[student number]" and upload it onto time and in the assignment "Practical Test 2" attached to Week 11.

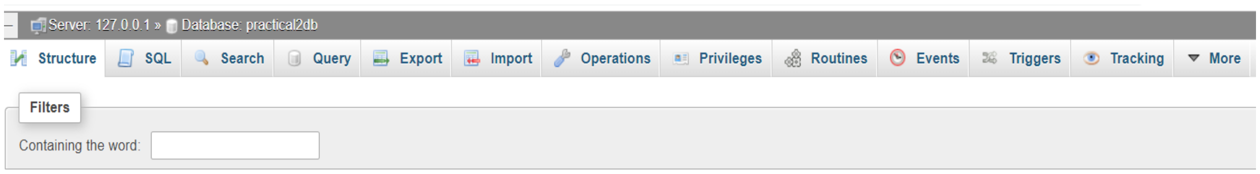
Task 1: Create a database and name it practical2DB. (5 marks)

SQL:

CREATE DATABASE practical2DB;

5

OUTPUT:



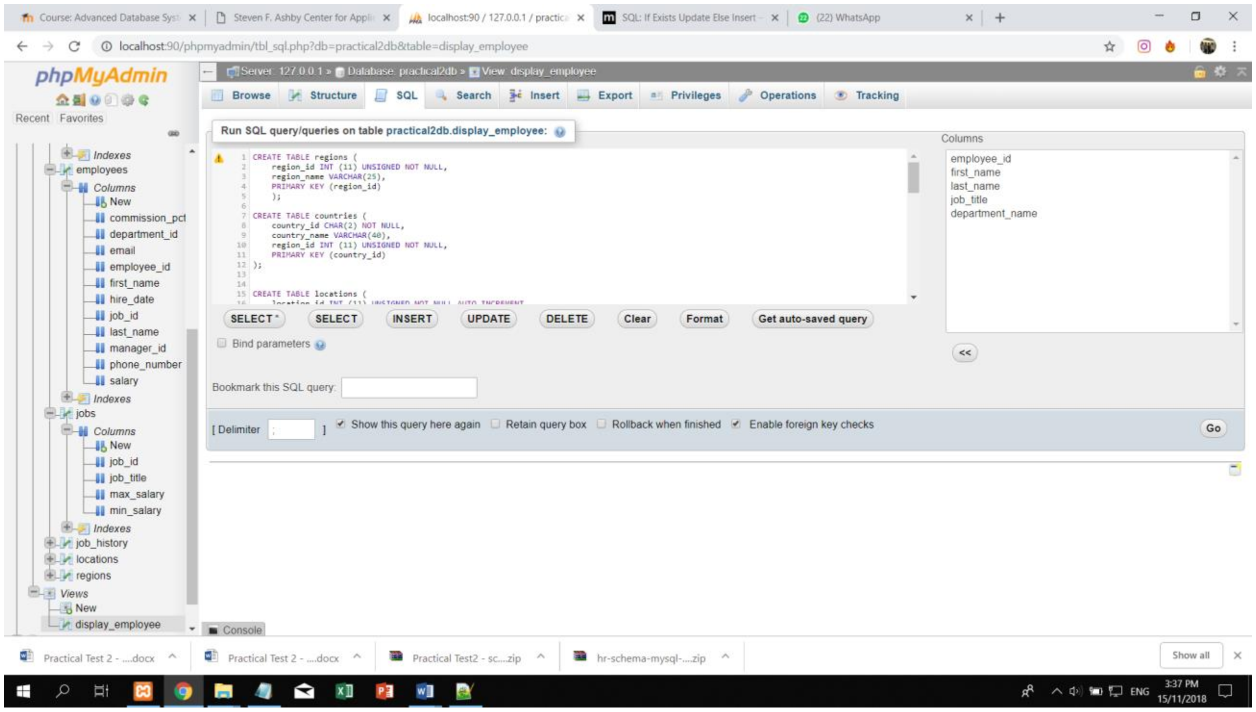
Task 2: Import all tables and records of each table available inside the scheme.

Screenshot 1: Inserting tables and their primary keys also their foreign keys

Screenshot 2: Inserting all data / records into the tables created

Screenshot 3: Display output of entire tables

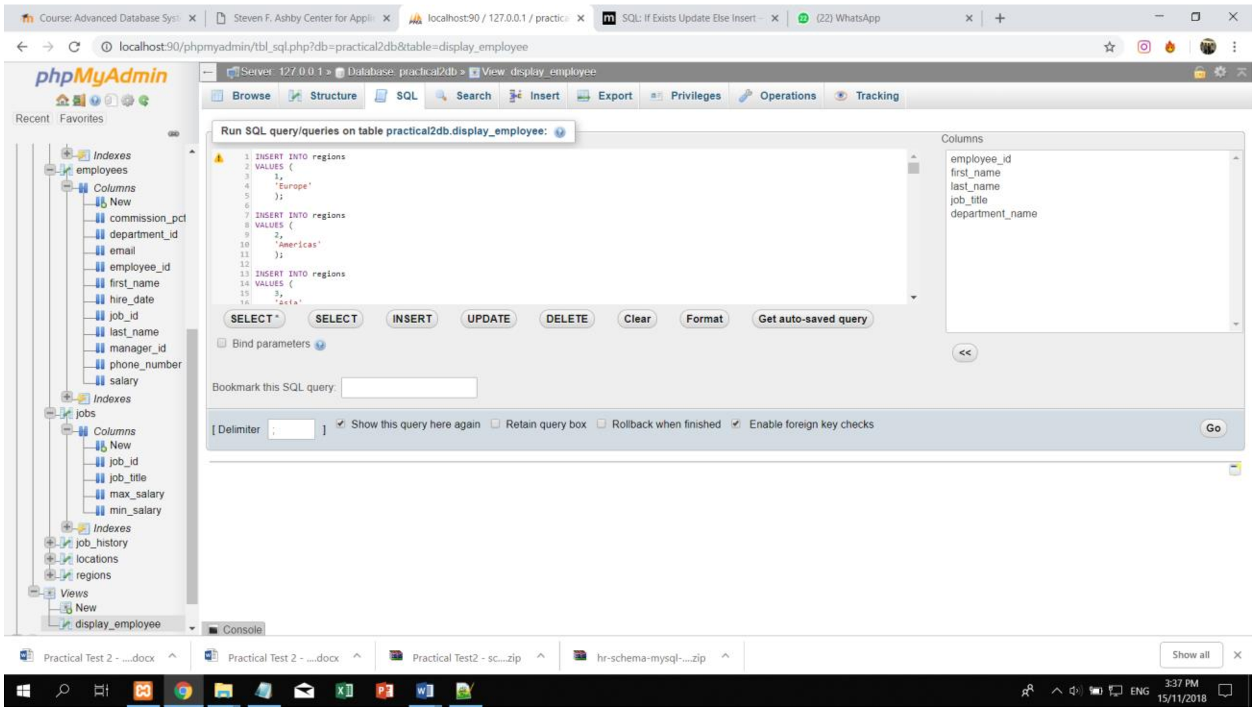
Screenshots are as displayed below



Run SQL query/queries on table practical2db.display_employee:

```
CREATE TABLE regions (
    region_id INT (11) UNSIGNED NOT NULL,
    region_name VARCHAR(25),
    PRIMARY KEY (region_id)
    );

CREATE TABLE countries (
    country_id CHAR(2) NOT NULL,
    country_name VARCHAR(40),
    region_id INT (11) UNSIGNED NOT NULL,
    PRIMARY KEY (country_id)
);


CREATE TABLE locations (
```

Screenshot 1: Inserting tables and their primary keys also their foreign keys via SQL



Run SQL query/queries on table practical2db.display_employee:

```
INSERT INTO regions
VALUES (
    1,
    'Europe'
    );

INSERT INTO regions
VALUES (
    2,
    'Americas'
    );

INSERT INTO regions
VALUES (
    3,
```

Screenshot 2: Inserting all data / records into the tables created via SQL

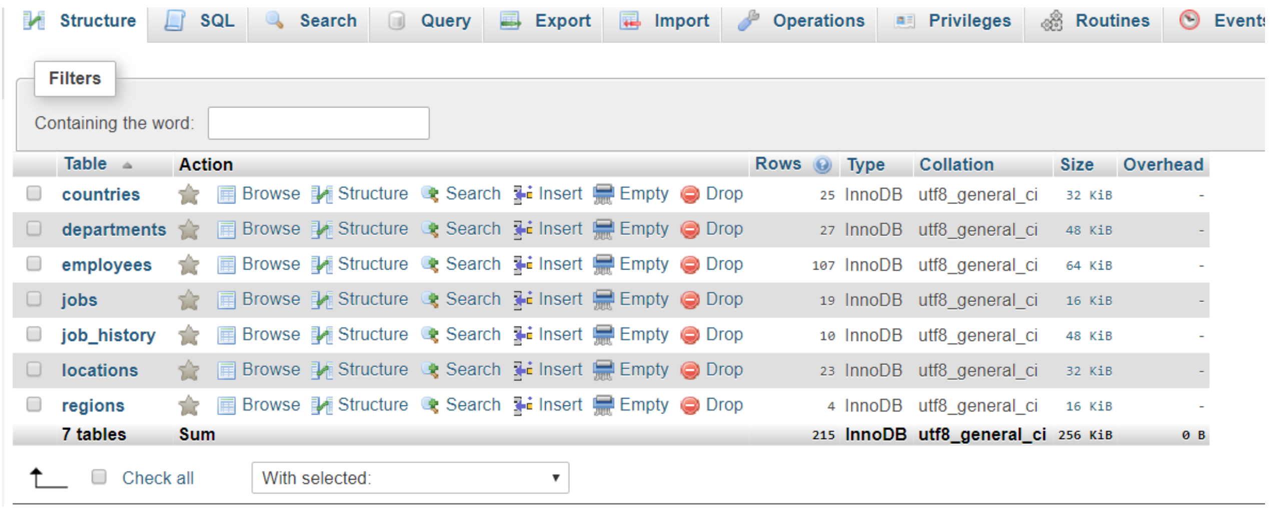


Table	Action	Rows	Type	Collation	Size	Overhead
countries	Browse Structure Search Insert Empty Drop	25	InnoDB	utf8_general_ci	32 KiB	-
departments	Browse Structure Search Insert Empty Drop	27	InnoDB	utf8_general_ci	48 KiB	-
employees	Browse Structure Search Insert Empty Drop	107	InnoDB	utf8_general_ci	64 KiB	-
jobs	Browse Structure Search Insert Empty Drop	19	InnoDB	utf8_general_ci	16 KiB	-
job_history	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8_general_ci	48 KiB	-
locations	Browse Structure Search Insert Empty Drop	23	InnoDB	utf8_general_ci	32 KiB	-
regions	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	16 KiB	-
7 tables	Sum	215	InnoDB	utf8_general_ci	256 KiB	0 B

Screenshot 3: Display output of entire tables via SQL

10

(10 marks)

Task3: Create a view listing all employees' name, employee ID, job title, and department name.

(15 marks)

SQL :

CREATE VIEW display_employee AS

SELECT employees.employee_id, employees.first_name, employees.last_name,

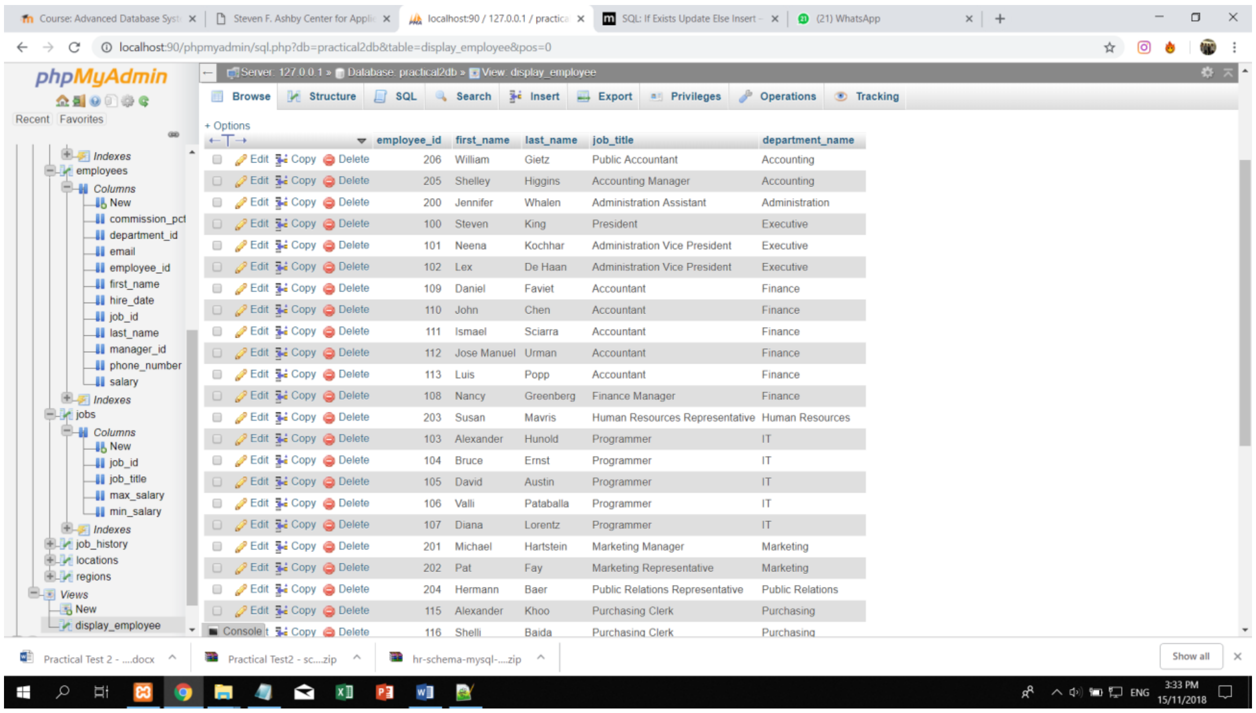
jobs.job_title, departments.department_name

FROM employees JOIN departments ON employees.department_id =

departments.department_id JOIN jobs ON jobs.job_id = employees.job_id;

OUTPUT:

15



localhost:90/phpmyadmin/sql.php?db=practical2db&table=display_employee&pos=0

Server: 127.0.0.1 » Database: practical2db » View: display_employee

Browse | Structure | SQL | Search | Insert | Export | Privileges | Operations | Tracking

+ Options

	employee_id	first_name	last_name	job_title	department_name
Edit Copy Delete	206	William	Gietz	Public Accountant	Accounting
Edit Copy Delete	205	Shelley	Higgins	Accounting Manager	Accounting
Edit Copy Delete	200	Jennifer	Whalen	Administration Assistant	Administration
Edit Copy Delete	100	Steven	King	President	Executive
Edit Copy Delete	101	Neena	Kochhar	Administration Vice President	Executive
Edit Copy Delete	102	Lex	De Haan	Administration Vice President	Executive
Edit Copy Delete	109	Daniel	Faviet	Accountant	Finance
Edit Copy Delete	110	John	Chen	Accountant	Finance
Edit Copy Delete	111	Ismael	Sciarra	Accountant	Finance
Edit Copy Delete	112	Jose Manuel	Urman	Accountant	Finance
Edit Copy Delete	113	Luis	Popp	Accountant	Finance
Edit Copy Delete	108	Nancy	Greenberg	Finance Manager	Finance
Edit Copy Delete	203	Susan	Mavris	Human Resources Representative	Human Resources
Edit Copy Delete	103	Alexander	Hunold	Programmer	IT
Edit Copy Delete	104	Bruce	Ernst	Programmer	IT
Edit Copy Delete	105	David	Austin	Programmer	IT
Edit Copy Delete	106	Valli	Pataballa	Programmer	IT
Edit Copy Delete	107	Diana	Lorentz	Programmer	IT
Edit Copy Delete	201	Michael	Hartstein	Marketing Manager	Marketing
Edit Copy Delete	202	Pat	Fay	Marketing Representative	Marketing
Edit Copy Delete	204	Hermann	Baer	Public Relations Representative	Public Relations
Edit Copy Delete	115	Alexander	Khoo	Purchasing Clerk	Purchasing
Copy Delete	116	Shelli	Baida	Purchasing Clerk	Purchasing

Task 4: Create a procedure to show a report that displays the full name and salary of employees who earn more than $7,000. Then Call the procedure and capture a screenshot of the results and embed at the bottom of your answer. (15 marks)

SQL:

DELIMITER //

CREATE PROCEDURE display_employee_above7k (IN param1 int)

BEGIN

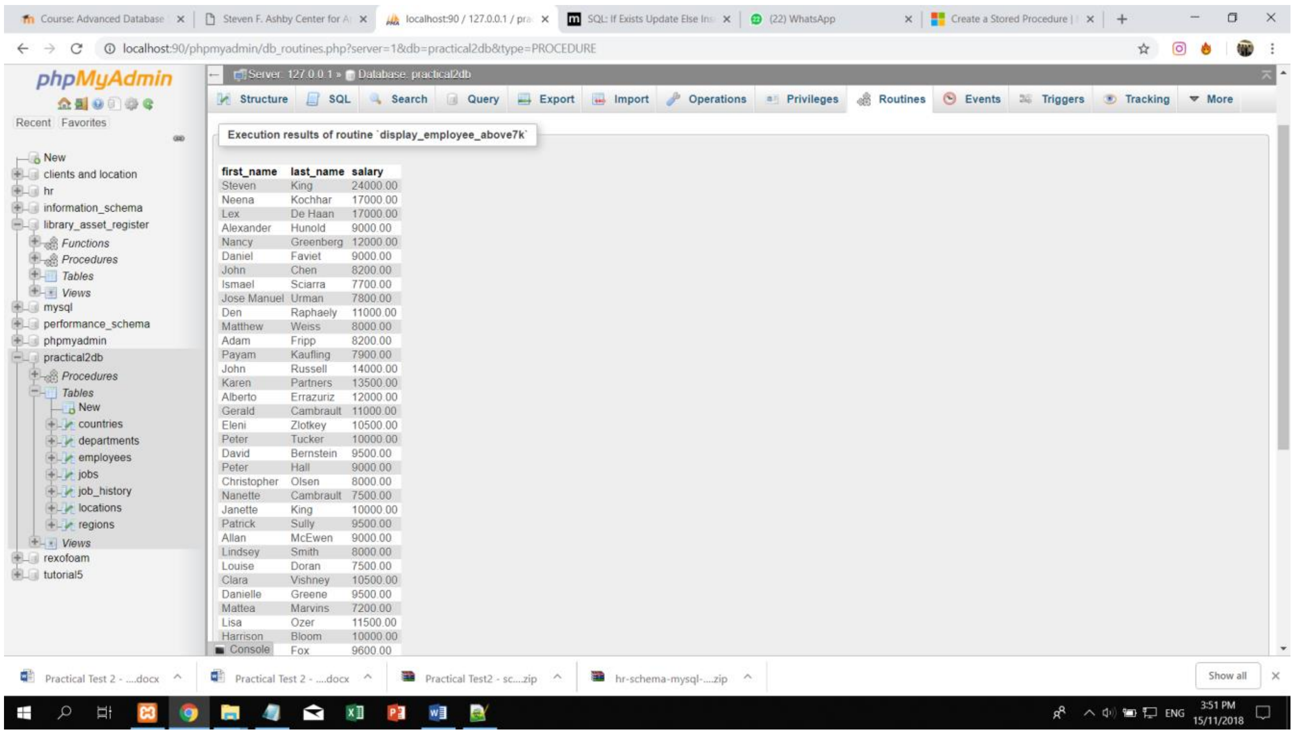
SELECT first_name, last_name, salary FROM `employees` WHERE salary > 7000;

end;//

DELIMITER ;

OUTPUT:

15



Execution results of routine `display_employee_above7k`

first_name	last_name	salary
Steven	King	24000.00
Neena	Kochhar	17000.00
Lex	De Haan	17000.00
Alexander	Hunold	9000.00
Nancy	Greenberg	12000.00
Daniel	Faviet	9000.00
John	Chen	8200.00
Ismael	Sciarra	7700.00
Jose Manuel	Urman	7800.00
Den	Raphaely	11000.00
Matthew	Weiss	8000.00
Adam	Fripp	8200.00
Payam	Kaufling	7900.00
John	Russell	14000.00
Karen	Partners	13500.00
Alberto	Errazuriz	12000.00
Gerald	Cambrault	11000.00
Eleni	Zlotkey	10500.00
Peter	Tucker	10000.00
David	Bernstein	9500.00
Peter	Hall	9000.00
Christopher	Olsen	8000.00
Nanette	Cambrault	7500.00
Janette	King	10000.00
Patrick	Sully	9500.00
Allan	McEwen	9000.00
Lindsey	Smith	8000.00
Louise	Doran	7500.00
Clara	Vishney	10500.00
Danielle	Greene	9500.00
Mattea	Marvins	7200.00
Lisa	Ozer	11500.00
Harrison	Bloom	10000.00
	Fox	9600.00

Task 5: Create a procedure to show a report that produces employee_id, full name and salary. The user should be allowed to key in the values of salary and department number. Consider the following items: (20 marks)

a) Using the IN operator for the department number (example: 10 and 20)

b) Using range selection for salary (condition)

Capture the screenshot of the results and embed at the bottom of your answer.

Justification user key in department_id as parameter 1, minimal salary as parameter 2 and maximum salary as parameter 3, procedure check department id and those with salary between the minimum and maximum entered.

Parameter format (department_id, minimum salary, maximum salary)

SQL:

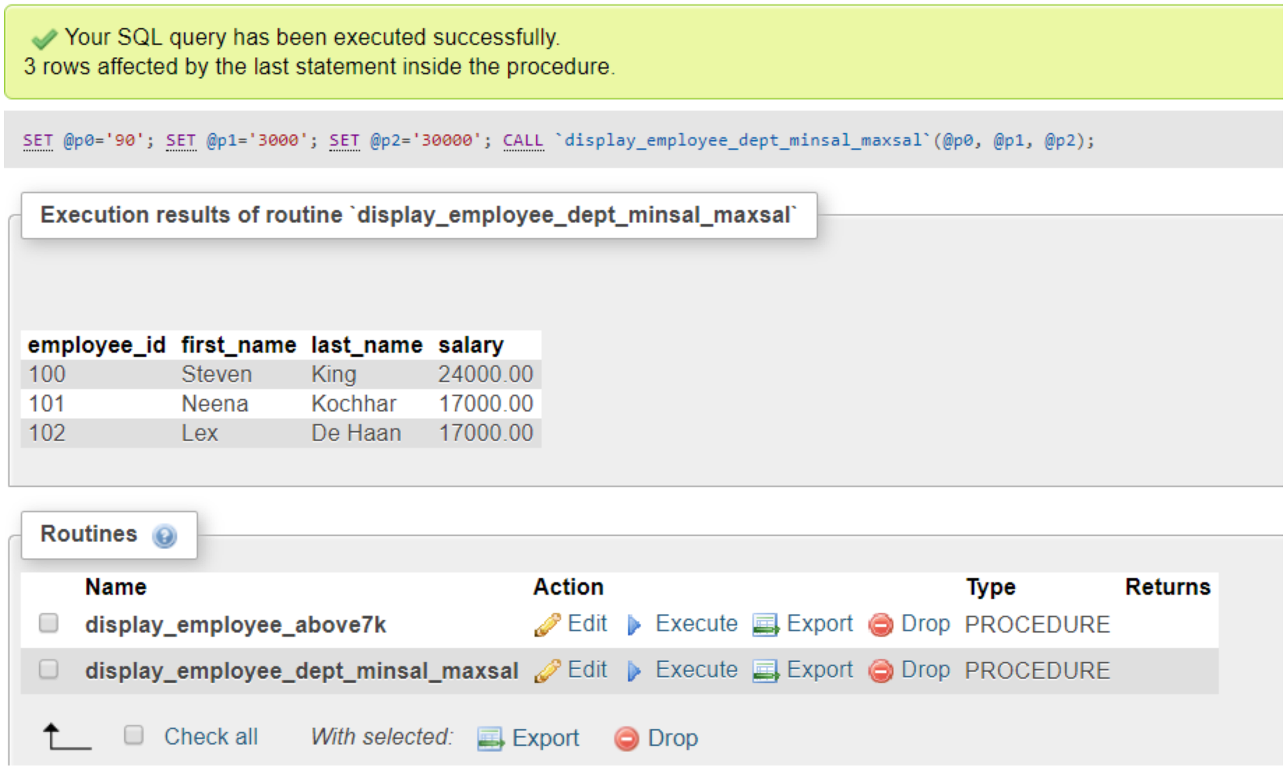
DELIMITER //

CREATE PROCEDURE display_employee_dept_minsal_maxsal (IN param1 int, IN param2 int, IN param3 int)

BEGIN

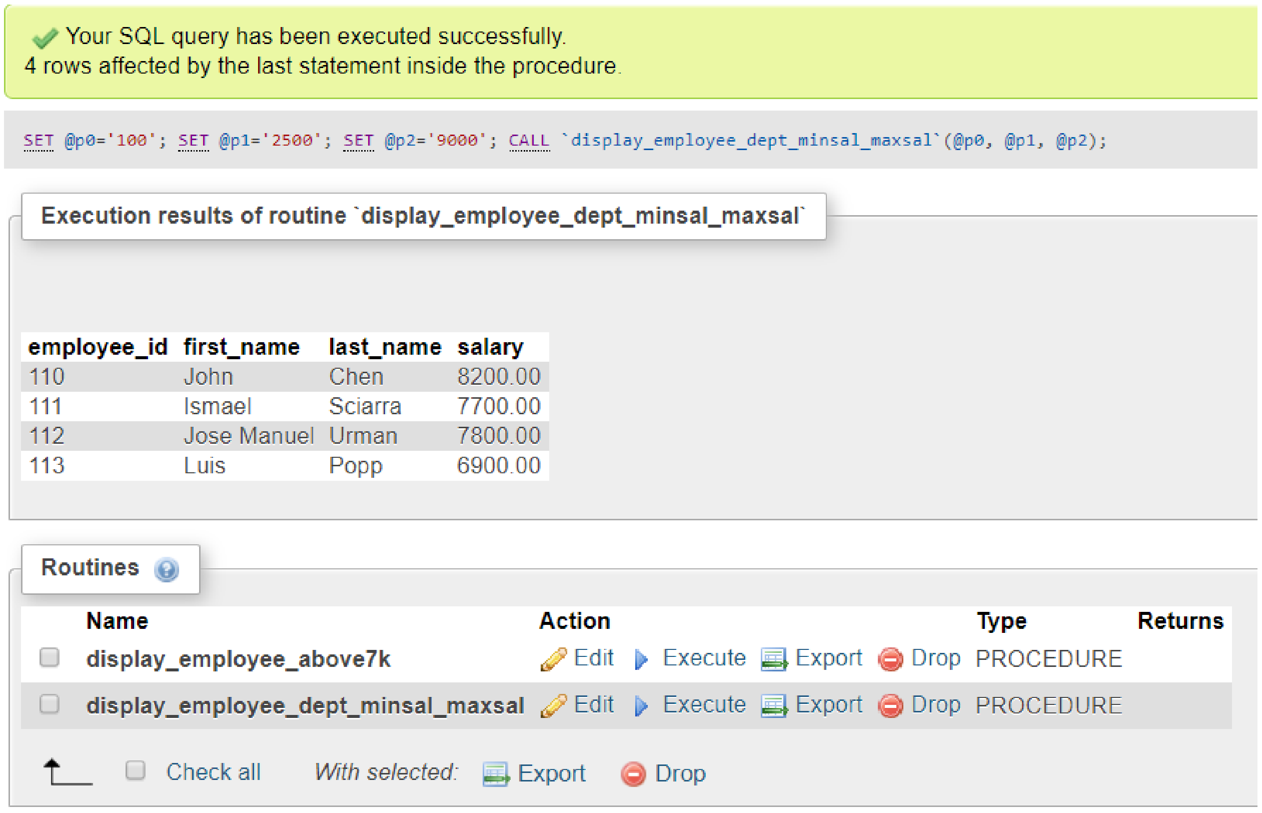
```
SELECT employee_id, first_name, last_name, salary
FROM employees
WHERE department_id = param1
AND salary > param2
AND salary < param3;
end;//
DELIMITER ;
```

OUTPUTS: 20



```
Your SQL query has been executed successfully.
3 rows affected by the last statement inside the procedure.

SET @p0='90'; SET @p1='3000'; SET @p2='30000'; CALL `display_employee_dept_minsal_maxsal`(@p0, @p1, @p2);

Execution results of routine `display_employee_dept_minsal_maxsal`

employee_id  first_name  last_name  salary
100          Steven      King       24000.00
101          Neena       Kochhar    17000.00
102          Lex         De Haan    17000.00

Routines

Name                                 Action                           Type       Returns
display_employee_above7k             Edit  Execute  Export  Drop      PROCEDURE
display_employee_dept_minsal_maxsal  Edit  Execute  Export  Drop      PROCEDURE

Check all   With selected:  Export  Drop
```

Executed using (90, 3000, 30000)



Your SQL query has been executed successfully.
4 rows affected by the last statement inside the procedure.

```
SET @p0='100'; SET @p1='2500'; SET @p2='9000'; CALL `display_employee_dept_minsal_maxsal`(@p0, @p1, @p2);
```

Execution results of routine `display_employee_dept_minsal_maxsal`

employee_id	first_name	last_name	salary
110	John	Chen	8200.00
111	Ismael	Sciarra	7700.00
112	Jose Manuel	Urman	7800.00
113	Luis	Popp	6900.00

Routines

Name	Action	Type	Returns
display_employee_above7k	Edit Execute Export Drop	PROCEDURE	
display_employee_dept_minsal_maxsal	Edit Execute Export Drop	PROCEDURE	

Check all With selected: Export Drop

Executed using (100, 2500, 9000)

Task 6: Create a function to produce a report that categorize staffs into 3 groups as Table 1 to classify them by the salary they earned. Then call the function and display the outcome. (20 marks)

Table 1: Classifications of staffs based on their salary

Salary less than 4,000	Low
Salary from 4,000 to 10,000	Medium
Salary above 10,000	High

SQL for function:

```
DELIMITER |

CREATE FUNCTION classify_salary (oldSal double)

RETURNS varchar(10)

DETERMINISTIC

BEGIN
```

```
        DECLARE sal_level varchar(15);
        if (salary < 4000) THEN
        SET sal_level = "Low";
    ELSEIF (salary < 10000) THEN
        SET sal_level = "Medium";
    ELSEIF (salary > 10000) THEN
        SET sal_level = "High";
    END IF;
    return sal_level;
END; |
```

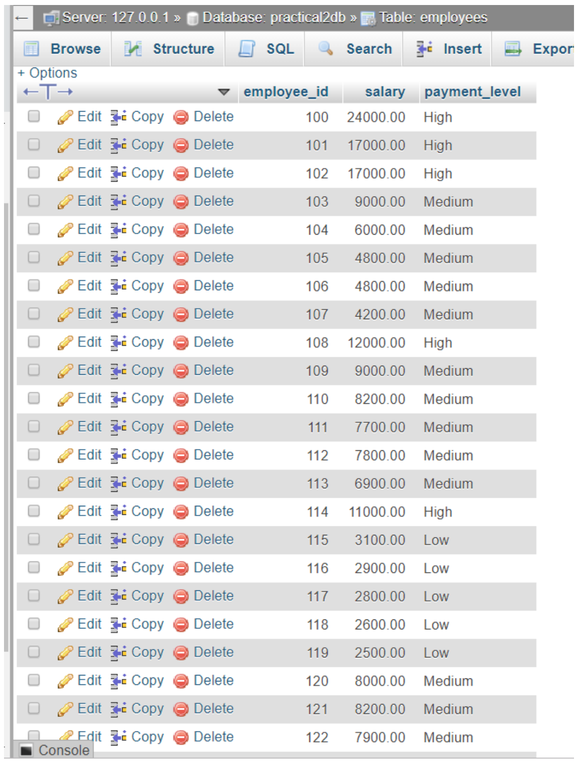
18

Justification of code: the first if measures those below 4000, when going to the next else if, the salary input is guaranteed to be more then 4000 but less then 10000

SQL to display report:

```
SELECT employee_id, salary, classify_salary(salary) as payment_level
FROM employees;
```

Output of Report:


Server: 127.0.0.1 » Database: practical2db » Table: employees
Browse
Structure
SQL
Search
Insert
Expor
+ Options
Console

	employee_id	salary	payment_level
Edit Copy Delete	100	24000.00	High
Edit Copy Delete	101	17000.00	High
Edit Copy Delete	102	17000.00	High
Edit Copy Delete	103	9000.00	Medium
Edit Copy Delete	104	6000.00	Medium
Edit Copy Delete	105	4800.00	Medium
Edit Copy Delete	106	4800.00	Medium
Edit Copy Delete	107	4200.00	Medium
Edit Copy Delete	108	12000.00	High
Edit Copy Delete	109	9000.00	Medium
Edit Copy Delete	110	8200.00	Medium
Edit Copy Delete	111	7700.00	Medium
Edit Copy Delete	112	7800.00	Medium
Edit Copy Delete	113	6900.00	Medium
Edit Copy Delete	114	11000.00	High
Edit Copy Delete	115	3100.00	Low
Edit Copy Delete	116	2900.00	Low
Edit Copy Delete	117	2800.00	Low
Edit Copy Delete	118	2600.00	Low
Edit Copy Delete	119	2500.00	Low
Edit Copy Delete	120	8000.00	Medium
Edit Copy Delete	121	8200.00	Medium
Edit Copy Delete	122	7900.00	Medium

Task 7: Create a function to raise salary of the staff by a given percentage (n%) and provide a report like Table 2. (Call the function for raise of 5% and display the results).

(20 marks)

Table 2: Raised Salaries

Name	Salary	Raised Salary
John	100000	105000
Mary	50000	52500

SQL for Function:

```
DELIMITER |

CREATE FUNCTION raise_salary (oldSal double, amount double)

RETURNS double

DETERMINISTIC

BEGIN

DECLARE raisedSal double;
```

Set raisedSal = oldSal * (1 + (amount/100)); return raisedSal;

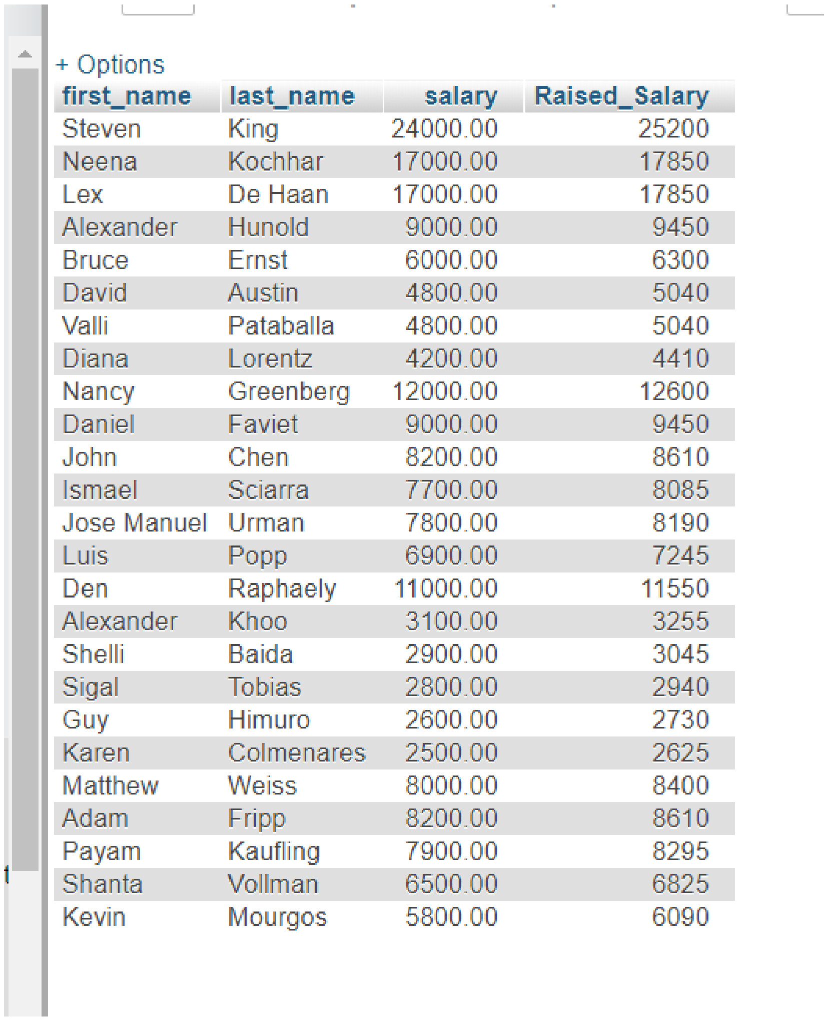
END;|

20

SQL call function:

SELECT first_name, last_name, salary, raise_salary(salary, 5) AS Raised_Salary FROM `employees` WHERE 1

Output:



+ Options

first_name	last_name	salary	Raised_Salary
Steven	King	24000.00	25200
Neena	Kochhar	17000.00	17850
Lex	De Haan	17000.00	17850
Alexander	Hunold	9000.00	9450
Bruce	Ernst	6000.00	6300
David	Austin	4800.00	5040
Valli	Pataballa	4800.00	5040
Diana	Lorentz	4200.00	4410
Nancy	Greenberg	12000.00	12600
Daniel	Faviet	9000.00	9450
John	Chen	8200.00	8610
Ismael	Sciarra	7700.00	8085
Jose Manuel	Urman	7800.00	8190
Luis	Popp	6900.00	7245
Den	Raphaely	11000.00	11550
Alexander	Khoo	3100.00	3255
Shelli	Baida	2900.00	3045
Sigal	Tobias	2800.00	2940
Guy	Himuro	2600.00	2730
Karen	Colmenares	2500.00	2625
Matthew	Weiss	8000.00	8400
Adam	Fripp	8200.00	8610
Payam	Kaufling	7900.00	8295
Shanta	Vollman	6500.00	6825
Kevin	Mourgos	5800.00	6090

Task 8: Create a table that shows list of departments and the total salary paid to staff of each department and then create a trigger to update the total salary of a department when a new employee is hired. (25 marks)

SQL create table:

20

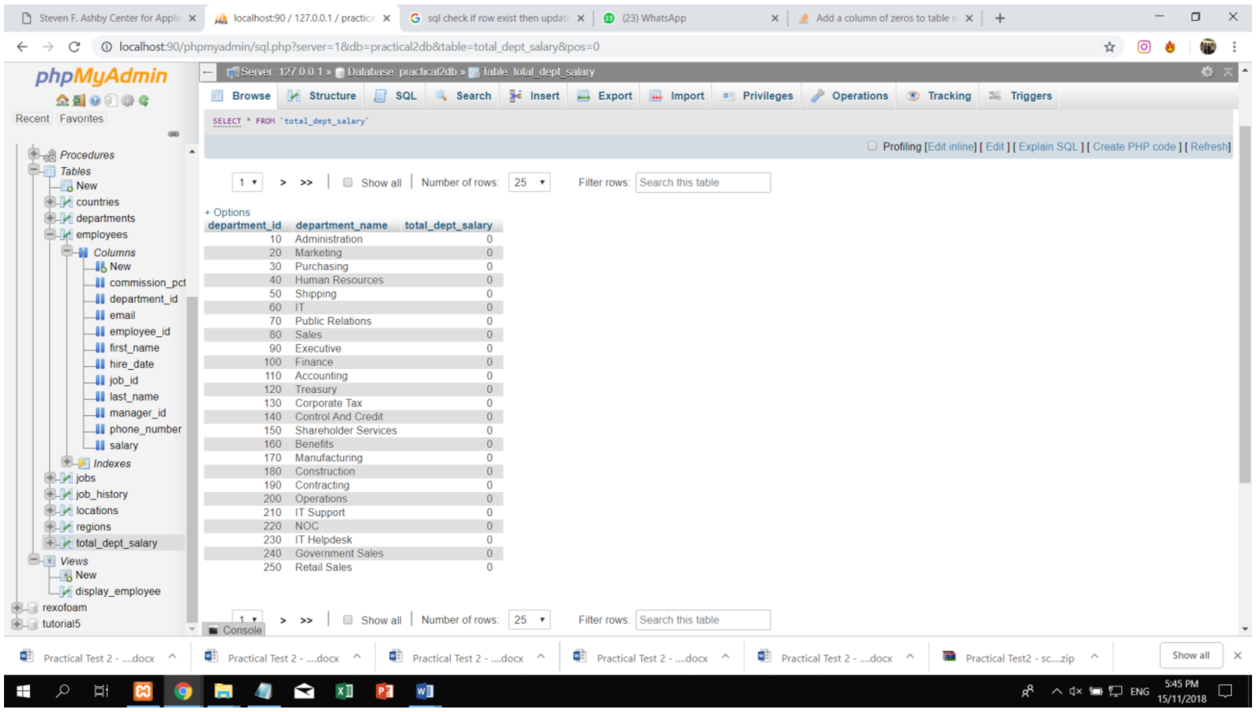
create table total_dept_salary_table AS

SELECT department_id,department_name, 0 as total_dept_salary

FROM departments

GROUP BY department_id ASC;

Output table:



```
SELECT * FROM `total_dept_salary`
```

department_id	department_name	total_dept_salary
10	Administration	0
20	Marketing	0
30	Purchasing	0
40	Human Resources	0
50	Shipping	0
60	IT	0
70	Public Relations	0
80	Sales	0
90	Executive	0
100	Finance	0
110	Accounting	0
120	Treasury	0
130	Corporate Tax	0
140	Control And Credit	0
150	Shareholder Services	0
160	Benefits	0
170	Manufacturing	0
180	Construction	0
190	Contracting	0
200	Operations	0
210	IT Support	0
220	NOC	0
230	IT Helpdesk	0
240	Government Sales	0
250	Retail Sales	0

SQL trigger:

DELIMITER |

create trigger update_salary

after INSERT on employees

```
for each ROW
begin
    update total_dept_salary_table
    SET total_dept_salary = total_dept_salary + new.salary;
    Where department_id = new.department_id;



    end;|
```

Total: 123 /130