Database Management System – CS422 DE

Lab 4 – Week 9

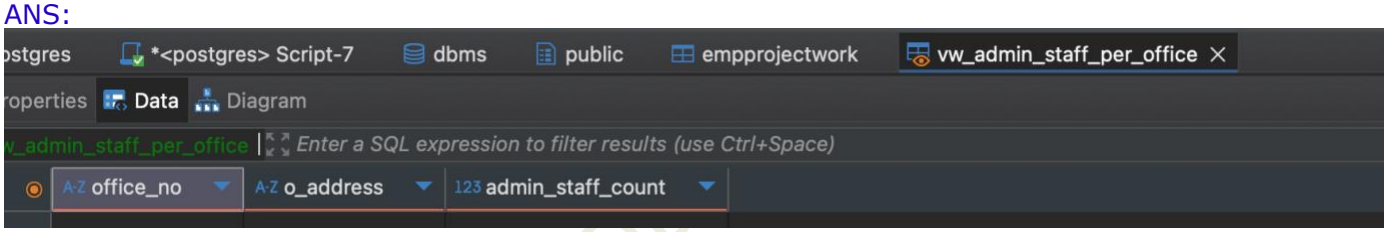
This Lab is based on Transact-SQL.

- Submit your *own work* on time. No credit will be given if the lab is submitted after the due date.
- Note that the completed lab should be submitted in **.pdf** format only.

Use the logical data model that you created in **Lab 3** (The *EasyDrive School of Motoring* case study), create tables in your selected DBMS, and put some data into them (populate them).

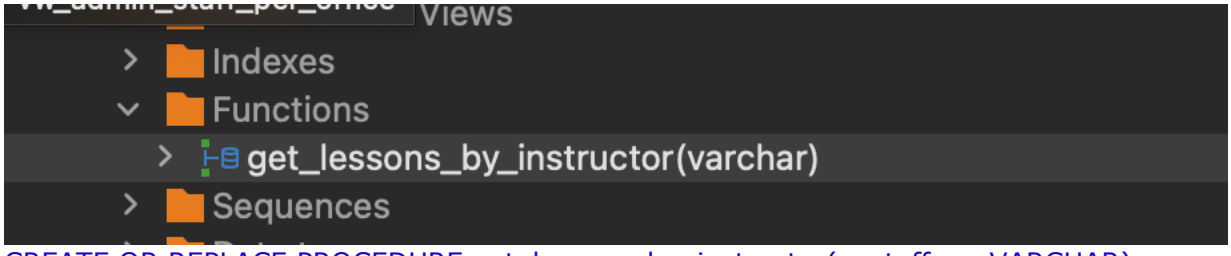
PREPARE TABLE SQL IS AT LAST PAGE

1) Create a View for the number of administrative staff located at each office.
ANS:




```
CREATE VIEW vw_admin_staff_per_office AS
SELECT
    o.office_no,
    o.o_address,
    COUNT(s.staff_no) AS admin_staff_count
FROM office o
LEFT JOIN staff s
    ON o.office_no = s.office_no
    AND s.job_title = 'Admin'
GROUP BY o.office_no, o.o_address;
```

2) Write a stored procedure that takes in one argument, the staff number of an instructor. The procedure outputs all details of all the lessons for that instructor.
ANS:




```
CREATE OR REPLACE PROCEDURE get_lessons_by_instructor(p_staff_no VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT *
    FROM lesson
    WHERE staff_no = p_staff_no;
END;
```

```
$$;
```

3) Create a trigger: In the Staff or Instructor table, add an attribute to keep track of the total number of clients that an instructor has. We will Not allow more than 20 clients per instructor. Add a trigger to Not allow a twenty-first client to be added.
ANS:

```
ALTER TABLE staff
ADD COLUMN total_clients INT DEFAULT 0;

CREATE OR REPLACE FUNCTION check_max_clients()
RETURNS TRIGGER AS $$
DECLARE
   client_count INT;
BEGIN
   SELECT COUNT(DISTINCT client_no)
   INTO client_count
   FROM lesson
   WHERE staff_no = NEW.staff_no;

   IF client_count >= 20 THEN
      RAISE EXCEPTION 'Instructor already has 20 clients';
   END IF;

   UPDATE staff
   SET total_clients = client_count + 1
   WHERE staff_no = NEW.staff_no;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_max_clients
BEFORE INSERT ON lesson
FOR EACH ROW
EXECUTE FUNCTION check_max_clients();
```


4) Cursors:
Use a cursor to read the rows of the Lesson table.
If the mileage for the lesson is over 20 miles, increase the fee by $5.
If the mileage for the lesson is over 25 miles, increase the fee by $8.
If the mileage for the lesson is over 30 miles, increase the fee by $10.

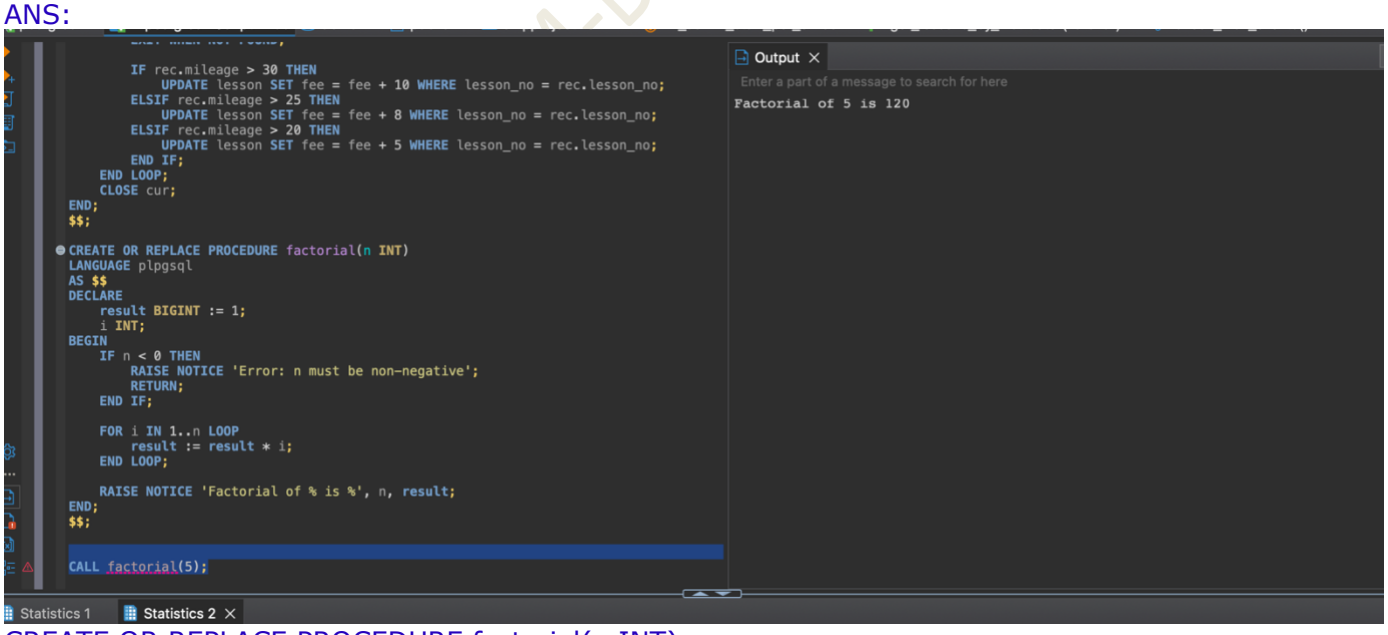
You may use an IF…ELSE…/CASE statement.
ANS:

```
ALTER TABLE lesson
ADD COLUMN fee DECIMAL(6,2) DEFAULT 50;

CREATE OR REPLACE PROCEDURE update_lesson_fees()
LANGUAGE plpgsql
```

```
AS $$
DECLARE
    rec RECORD;
    cur CURSOR FOR
        SELECT lesson_no, (mileage_finish - mileage_start) AS mileage
        FROM lesson;
BEGIN
    OPEN cur;
    LOOP
        FETCH cur INTO rec;
        EXIT WHEN NOT FOUND;

        IF rec.mileage > 30 THEN
            UPDATE lesson SET fee = fee + 10 WHERE lesson_no = rec.lesson_no;
        ELSIF rec.mileage > 25 THEN
            UPDATE lesson SET fee = fee + 8 WHERE lesson_no = rec.lesson_no;
        ELSIF rec.mileage > 20 THEN
            UPDATE lesson SET fee = fee + 5 WHERE lesson_no = rec.lesson_no;
        END IF;
    END LOOP;
    CLOSE cur;
END;
$$;
```

5) Write and execute a T-SQL stored procedure *Factorial*(*n*), which computes and outputs the factorial of the input parameter *n*. If *n* is negative, then the procedure prints an error message.

ANS:




```
CREATE OR REPLACE PROCEDURE factorial(n INT)
LANGUAGE plpgsql
AS $$
DECLARE
    result BIGINT := 1;
    i INT;
BEGIN
    IF n < 0 THEN
        RAISE NOTICE 'Error: n must be non-negative';
```

```
        RETURN;
    END IF;

    FOR i IN 1..n LOOP
        result := result * i;
    END LOOP;

    RAISE NOTICE 'Factorial of % is %', n, result;
END;
$$;
CALL factorial(5);
```

Prepare tables SQL

```
CREATE TABLE office (
    office_no       VARCHAR(10) PRIMARY KEY,
    o_address       VARCHAR(100),
    o_postcode      VARCHAR(10),
    o_tel_no        VARCHAR(15),
    o_fax_no        VARCHAR(15)
);

CREATE TABLE staff (
    staff_no    VARCHAR(10) PRIMARY KEY,
    f_name      VARCHAR(50),
    l_name      VARCHAR(50),
    s_address   VARCHAR(100),
    job_title   VARCHAR(30),
    salary      DECIMAL(8,2),
    nin         VARCHAR(20),
    sex         CHAR(1),
    dob         DATE,
    office_no   VARCHAR(10),
    CONSTRAINT fk_staff_office
        FOREIGN KEY (office_no)
        REFERENCES office(office_no)
);

CREATE TABLE vehicle (
    veh_reg_no  VARCHAR(15) PRIMARY KEY,
    model       VARCHAR(30),
    make        VARCHAR(30),
    color       VARCHAR(20),
    capacity    INT
);
CREATE TABLE client (
    client_no      VARCHAR(10) PRIMARY KEY,
    c_address      VARCHAR(100),
    c_postcode     VARCHAR(10),
    c_tel_no       VARCHAR(15),
    d_license_no   VARCHAR(20),
    sex            CHAR(1),
    dob            DATE
```

```
);

CREATE TABLE lesson (
    lesson_no      VARCHAR(10) PRIMARY KEY,
    lesson_date    DATE,
    lesson_time    TIME,
    stage          VARCHAR(20),
    progress       VARCHAR(50),
    comments       TEXT,
    mileage_start  INT,
    mileage_finish INT,
    staff_no       VARCHAR(10),
    client_no      VARCHAR(10),
    veh_reg_no     VARCHAR(15),
    CONSTRAINT fk_lesson_staff
        FOREIGN KEY (staff_no)
        REFERENCES staff(staff_no),
    CONSTRAINT fk_lesson_client
        FOREIGN KEY (client_no)
        REFERENCES client(client_no),
    CONSTRAINT fk_lesson_vehicle
        FOREIGN KEY (veh_reg_no)
        REFERENCES vehicle(veh_reg_no)
);
CREATE TABLE inspection (
    inspection_id SERIAL PRIMARY KEY,
    insp_date     DATE,
    insp_time     TIME,
    faults_found  TEXT,
    comments      TEXT,
    staff_no      VARCHAR(10),
    veh_reg_no    VARCHAR(15),
    CONSTRAINT fk_inspection_staff
        FOREIGN KEY (staff_no)
        REFERENCES staff(staff_no),
    CONSTRAINT fk_inspection_vehicle
        FOREIGN KEY (veh_reg_no)
        REFERENCES vehicle(veh_reg_no)
);


CREATE TABLE interview (
    interview_id SERIAL PRIMARY KEY,
    i_date       DATE,
    i_time       TIME,
    i_room       VARCHAR(20),
    i_comments   TEXT,
    d_license_no VARCHAR(20),
    staff_no     VARCHAR(10),
    client_no    VARCHAR(10),
    CONSTRAINT fk_interview_staff
        FOREIGN KEY (staff_no)
        REFERENCES staff(staff_no),
    CONSTRAINT fk_interview_client
        FOREIGN KEY (client_no)
        REFERENCES client(client_no)
);
```

```
CREATE TABLE driving_test (
   test_no        VARCHAR(10) PRIMARY KEY,
   test_date       DATE,
   test_time       TIME,
   test_centre     VARCHAR(50),
   tester_name     VARCHAR(50),
   attempt         INT,
   result         VARCHAR(10),
   p_test_comment TEXT,
   t_test_comment TEXT,
   client_no       VARCHAR(10),
   CONSTRAINT fk_test_client
      FOREIGN KEY (client_no)
      REFERENCES client(client_no)
);
```