|  |
| --- |
| ------------------------------------------------- |
|  | --Here starts the solution to the points 4 to 8-- |
|  | ------------------------------------------------- |
|  |  |
|  | /\*Point 4. Create a view in which can be seen Registration\_number, Mileage, Model, Current\_value, class, Replacement\_Value, Status, Required\_special\_qualification, Number\_of\_services, |
|  | Total\_money\_spent\_in\_repair and should be only run by users associated to "finance\_profile?"\*/ |
|  | CREATE OR REPLACE VIEW seeing\_details AS |
|  | SELECT vehicles.registration\_number, |
|  | vehicles.mileage, |
|  | vehicles.value\_Ve, |
|  | vehicles.model\_Ve, |
|  | type\_of\_vehicles.description, |
|  | vehicles.replacement\_val, |
|  | vehicles.status, |
|  | vehicles.requires\_qualification, |
|  | COALESCE(aggregated\_services.service\_count, 0) as service\_count, |
|  | COALESCE(aggregated\_costs.total\_paid, 0) as total\_paid |
|  | FROM vehicles JOIN type\_of\_vehicles on type\_of\_vehicles.id\_type = vehicles.id\_type |
|  |  |
|  | LEFT JOIN (select service\_details.id\_vehicles, |
|  | COUNT(service\_details.id\_service) AS service\_count |
|  |  |
|  | FROM service\_details GROUP BY service\_details.id\_vehicles) aggregated\_services |
|  | ON aggregated\_services.id\_vehicles = vehicles.id\_vehicle |
|  |  |
|  | LEFT JOIN (select repair\_costs.id\_vehicles, SUM(repair\_cost) AS total\_paid |
|  |  |
|  | FROM repair\_costs GROUP BY repair\_costs.id\_vehicles) aggregated\_costs |
|  |  |
|  | ON aggregated\_costs.id\_vehicles = vehicles.id\_vehicle |
|  | WHERE vehicles.id\_type IN (4, 5, 6, 7, 8) AND vehicles.bought\_date >= sysdate - interval '2' year |
|  |  |
|  | ORDER BY vehicles.mileage DESC; |
|  |  |
|  | GRANT SELECT ON seeing\_details TO second\_user; |
|  |  |
|  | /\*Point 5. Create a function to assign the name of a service, it should return a string.If a name already exists for the input code of vehicle on service\_details then the returned string shall be null, |
|  | otherwise the string shall return a string that says 3000\_service, 6000\_service and 9000\_service according to the mileage\*/ |
|  |  |
|  | CREATE OR REPLACE FUNCTION assign\_name (vehicle\_in IN integer) |
|  | RETURN varchar2 AS |
|  | mileage\_service VARCHAR2(20); |
|  | mileageda number := 0; |
|  | is\_found number := 0; |
|  | BEGIN |
|  | --SELECT id\_vehicles INTO is\_found FROM service\_details |
|  | -- where id\_vehicles = vehicle\_in; |
|  |  |
|  | --if sql%found then |
|  | --mileage\_service :=''; |
|  | --else |
|  | SELECT mileage INTO mileageda FROM vehicles where id\_vehicle = vehicle\_in; |
|  |  |
|  | if mileageda >= 0 AND mileageda <= 3000 then |
|  | mileage\_service := '3000\_service'; |
|  | else |
|  | if mileageda >3000 AND mileageda <=6000 then |
|  | mileage\_service := '6000\_service'; |
|  | else |
|  | mileage\_service := '9000\_service'; |
|  | end if; |
|  | end if; |
|  | --end if; |
|  | RETURN mileage\_service; |
|  | END; |
|  |  |
|  |  |
|  | SELECT assign\_name(2) FROM DUAL; |
|  |  |
|  |  |
|  | /\*Point 6. Create a stored procedure which accepts the id of vehicle as argument, inside the procedure you should: Call the function to assign the name of the service, |
|  | if the string is not empty, create a new service for that vehicle in status "Pending", current date, the name field should contain text "Mandatory 'X000\_service', description: current mileage. |
|  | If the string is empty do nothing.\*/ |
|  |  |
|  | CREATE OR REPLACE PROCEDURE REMOVE\_LOCATION (vehicle\_in IN number) AS |
|  | name\_service varchar(50); |
|  | mileagenew int; |
|  | pointer\_id number := service\_details\_sequence.nextval; |
|  | BEGIN |
|  | name\_service := assign\_name(vehicle\_in); |
|  | SELECT mileage INTO mileagenew FROM vehicles where id\_vehicle = vehicle\_in; |
|  | if name\_service is not null then |
|  |  |
|  | INSERT INTO service\_details VALUES (pointer\_id, vehicle\_in,'Mandatory '||name\_service, 'current mileage: '||mileagenew,SYSDATE,'pending'); |
|  |  |
|  | INSERT INTO service VALUES (pointer\_id, 1); |
|  | INSERT INTO service VALUES (pointer\_id, 2); |
|  | INSERT INTO service VALUES (pointer\_id, 3); |
|  | INSERT INTO service VALUES (pointer\_id, 4); |
|  | INSERT INTO service VALUES (pointer\_id, 5); |
|  | INSERT INTO service VALUES (pointer\_id, 6); |
|  | INSERT INTO service VALUES (pointer\_id, 7); |
|  | INSERT INTO service VALUES (pointer\_id, 8); |
|  | INSERT INTO service VALUES (pointer\_id, 9); |
|  | INSERT INTO service VALUES (pointer\_id, 10); |
|  | INSERT INTO service VALUES (pointer\_id, 11); |
|  | INSERT INTO service VALUES (pointer\_id, 12); |
|  | INSERT INTO service VALUES (pointer\_id, 13); |
|  | INSERT INTO service VALUES (pointer\_id, 14); |
|  | INSERT INTO service VALUES (pointer\_id, 15); |
|  | INSERT INTO service VALUES (pointer\_id, 16); |
|  | INSERT INTO service VALUES (pointer\_id, 17); |
|  | INSERT INTO service VALUES (pointer\_id, 18); |
|  | INSERT INTO service VALUES (pointer\_id, 19); |
|  |  |
|  | end if; |
|  | END; |
|  |  |
|  | --Point 7. Create a trigger which once the mileage column is updated, calls the procedure that's just been created. |
|  |  |
|  | CREATE OR REPLACE TRIGGER update\_mileage |
|  | AFTER UPDATE OF mileage ON vehicles |
|  | FOR EACH ROW |
|  |  |
|  | BEGIN |
|  | REMOVE\_LOCATION(:new.id\_vehicle); |
|  | END; |
|  |  |
|  |  |
|  | /\*Point 8. Create a stored procedure to decrease the current value of vehicles as follows: |
|  | a. If the class of vehicle is Moped, Motorcycle and Trike motorcycle 2% of the current value. |
|  | b. If the class of vehicle is Car, Light rigid heavy vehicle, Medium rigid heavy vehicle 3% of the current value. |
|  | c. Otherwise 5% of the current value. |
|  | d. Only "manager\_profile?" is allowed to run the procedure and should print at the end "Number of vehicles updated successfully: xx\*/ |
|  |  |
|  | CREATE OR REPLACE PROCEDURE decrease\_mileage AS |
|  | counter\_success number :=0; |
|  | BEGIN |
|  | UPDATE vehicles SET mileage = (mileage\*0.98) |
|  | WHERE id\_type IN (1, 2, 3); |
|  | counter\_success := sql%rowcount; |
|  |  |
|  | UPDATE vehicles SET mileage = (mileage\*0.97) |
|  | WHERE id\_type IN (4, 5, 6, 7); |
|  | counter\_success := counter\_success + sql%rowcount; |
|  |  |
|  | UPDATE vehicles SET mileage = (mileage\*0.95) |
|  | WHERE id\_type IN (8, 9); |
|  | counter\_success := counter\_success + sql%rowcount; |
|  |  |
|  | dbms\_output.put\_line('there were updated: ' || decrease\_mileage || 'registers'); |
|  |  |
|  | END; |
|  |  |
|  |  |
|  | BEGIN decrease\_mileage ; END; |
|  |  |
|  | GRANT EXECUTE ON decrease\_mileage TO first\_user; |