There is a mid-size restaurant with Mediterranean food of which the 70% of income is generated from food delivery. The Owner does not use any external delivery services but instead decided to lease a few small cars and hire drivers.

Due to the 2-year contracts with the Lessor, the Owner of the restaurant needs to fulfil certain conditions to avoid unnecessary financial impacts. For example one of the conditions is that each car should have a technical review and maintenance every 15,000 km. The Owner is not able to keep track of the car kilometer counters himself so he needs an information system in which all required information will be collected by his drivers on a daily basis. The system will also have a few other useful features to help him manage his cars fleet.

The other issue or need that this system will address is to make sure that the information about his business, which he wants to present on the restaurant website, is always up-to-date. For example he wants to show such information as the current number of cars in use, the number of employees and the number of restaurant locations.

My project contains the following objects:

TABLES:

1. Restaurant - contains the main information about the business;
2. Location - contains the address and contact details of all active and inactive locations of the Restaurant;
3. Employee - contains the list of all active and inactive employees, i.e. name, role, contract type, start and end of the contract, salary and the location id to specify the employee's actual workplace;
4. Car - contains the list of all active and inactive cars;
5. CarMileage - contains the information about value of the kilometer counter recorded on a daily basis by all drivers;
6. CarHistory - contains the history of events/task/actions related to company's cars;
7. ActionType - dictionary table, contains the list of possible actions or events or tasks that can be recorded in the CarHistory table;
8. CarHistoryStatus - dictionary table, contains the list of possible statuses for actions recorded in CarHistory table;
9. Role - dictionary table, contains the list of possible roles that an employee can be assigned;
10. ContractType - dictionary table, contains the list of possible contract types with employees;
11. PaymentCycle - dictionary table, contains the list of possible payment cycles that can be applied for salaries;
12. Status - dictionary table, contains the list of possible statuses (or states) related to car, employees and locations;
13. ActivityLog - contains records of all insert, update or delete actions performed by users of this system, in order to know who, when and in which table changed the data.

PROCEDURES:

1. p_UpdateActivityLog - it adds a new line to 'ActivityLog' table any time when there is an insert, update or delete action done by any user on any table. This procedure is executed by multiple triggers.
2. p_UpdateRestaurantData - calculates the number of cars, employees and locations of a particular restaurant recorded in 'Restaurant' table.
    a. In MS SQL the procedure is executed by triggers on tables 'Car', 'Employee' and 'Location' any time when insert or delete is done on these tables.
    b. In PL/SQL the procedure can be executed manually by providing the RestID value as a parameter.
3. p_DisplayAllEmpWithGivenRole - finds all employees with a specific role provided as parameter and prints their Employee IDs, Names and Location address. The cursor fetches data from table 'Employee' joined with table 'Location' where an employee has the specified role.

TRIGGERS:

1. t_UpdateActivityLog_[Table name] - all tables except for the 'ActivityLog' have a trigger which determines the activity type done on this table (i.e. insert, update or delete) and pass it together with the name of the source table to the procedure 'p_UpdateActivityLog', in order to create activity log record with date/time stamp, name of the current user, table name and action type.
2. t_Insert_CarHistory - it is an 'AFTER INSERT FOR EACH ROW' trigger on table 'CarMileage'. Its role is to monitor company's cars counters and help the Owner of the restaurant remember about regular technical inspections which are obligatory based on the lease contracts. CarMileage table is updated by company's drivers on a daily basis.

    The trigger checks the current value of the car counter (from 'CarMileage' table), the counter cycle after which the technical inspection should be done in case of this specific car (from 'Car' table), and adds a record into table 'CarHistory' when the value of the car counter is reaching the level of the required technical inspection cycle, in order to create a task/action to make a reservation at the car repair/service center for technical inspection visit.

    For example, if the technical inspections should be done after each 15,000 km and the counter cycle offset is set to 500 km, then:

    a. The first record (task/action) with status 'In progress' will be added to 'CarHistory' table when the car counter exceeds 14,500 km, e.g. 14,561 km;
    b. Between 14,561 km and 15,561 km, i.e. within 1,000 km (= counter cycle offset * 2) the system will allow for adding new records to 'CarMileage' table without creating another task/action into 'CarHistory' table;
    c. When car counter exceeds 15561 km and the status of the action previously recorded in 'CarHistory' is still 'In progress' then the trigger will add another record to 'CarHistory' with the same task/action and the current value of the car counter.

The trigger will continue to do b. and c. until the status on the latest task/action record will be changed manually to ‘Completed’.

d. After changing the status of the latest task/action record to ‘Completed’ the next one will be created when the current car counter value will be `reaching the level of the next technical inspection cycle, counted by adding 14500 km to the value of counter from the latest ‘CarHistory’ record with status ‘Completed’.`

The above scenarios can be tested in MS SQL using the test cases from file “5.3. NB-MSSql - Testing trigger 't_Insert_CarHistory'.sql”.