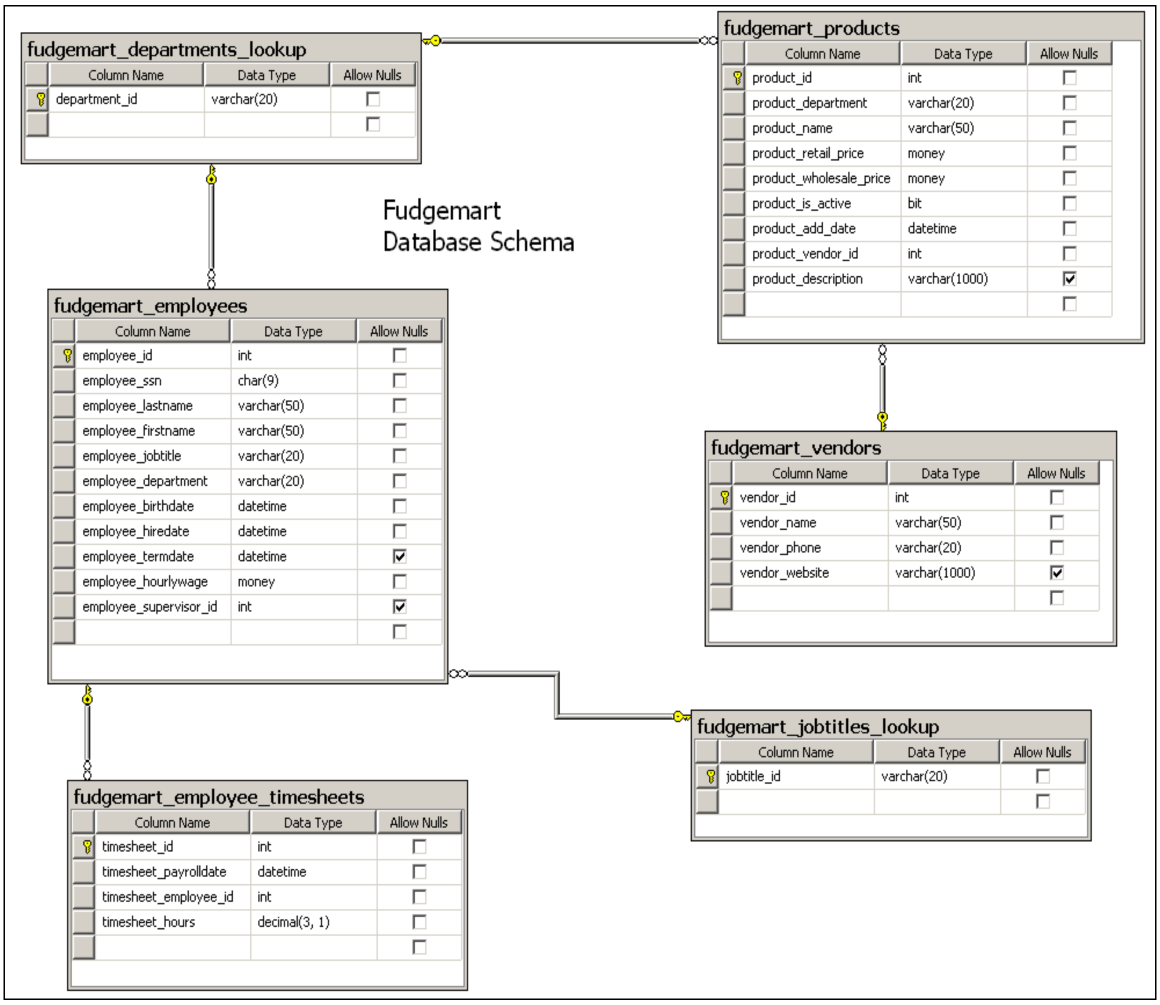
Given the Fudgemart database with the below diagram



fudgemart_departments_lookup

Column Name	Data Type	Allow Nulls
department_id	varchar(20)	☐
		☐

fudgemart_products

Column Name	Data Type	Allow Nulls
product_id	int	☐
product_department	varchar(20)	☐
product_name	varchar(50)	☐
product_retail_price	money	☐
product_wholesale_price	money	☐
product_is_active	bit	☐
product_add_date	datetime	☐
product_vendor_id	int	☐
product_description	varchar(1000)	☑
		☐

fudgemart_employees

Column Name	Data Type	Allow Nulls
employee_id	int	☐
employee_ssn	char(9)	☐
employee_lastname	varchar(50)	☐
employee_firstname	varchar(50)	☐
employee_jobtitle	varchar(20)	☐
employee_department	varchar(20)	☐
employee_birthdate	datetime	☐
employee_hiredate	datetime	☐
employee_termdate	datetime	☑
employee_hourlywage	money	☐
employee_supervisor_id	int	☑
		☐

fudgemart_vendors

Column Name	Data Type	Allow Nulls
vendor_id	int	☐
vendor_name	varchar(50)	☐
vendor_phone	varchar(20)	☐
vendor_website	varchar(1000)	☑
		☐

fudgemart_jobtitles_lookup

Column Name	Data Type	Allow Nulls
jobtitle_id	varchar(20)	☐
		☐

fudgemart_employee_timesheets

Column Name	Data Type	Allow Nulls
timesheet_id	int	☐
timesheet_payrolldate	datetime	☐
timesheet_employee_id	int	☐
timesheet_hours	decimal(3, 1)	☐
		☐

Preparation: Execute scripts in the sql.txt to create database with the name fudgemart_yourstudentcode.

Examination requirements (TOTAL 12 point):

PART 1 (Required). 60 minutes (5 Point)

1. Create an additional table named as ***fudgemart_agents*** described as below (2 point):

Column name	Description	Data type	Constraints	Other requirements
agent_id	Id of agent	Int	Primary key (0.25 point)	Automatically generated (identity) (0.25 point)
agent_name	Name of agent	Varchar(50)		
agent_address	Address of agent	Varchar(50)		
agent_country	Country of agent location in abbreviation	Varchar(2)		This field is indexed for improving query performance on agent_country (0.5 point)

agent_created_date		Datetime		The value cannot be greater than the current date (0.25 point)
agent_manager_id	The id of employee who is now sale manager at the agent	Int	Foreign key constraint, refer to the table fudgemart_employees (0.25 point)	
agent_status	Active or inactive	Bit	Default value = 1 (0.25 point)	Not null

2. Generate database diagram (including the additional table) and copy into the answer sheet (0.5 point)
3. Write Select statements (1.5 point)
   a. List all employees who work at the department Electronics and have the age greater than 30 at the present
   b. List all active products which have the whole sale price equals a half of retail price.
4. Create View containing (1 point):
   a. The list of active products provided by the vendor Mikey
   b. The list of payroll times in which the number of employees, who have hours less than 40, is greater than 2 people.

PART 2. 90 MINUTES (7 Point)

5. Stored procedure (4 point)
   a. (1.25 point) Write a stored procedure (named as ***p_fudgemart_add_new_vendor***) which is used to add a new vendor into the table fudgemart_vendors with the below requirement:
      - Check the vendor_id and vendor_name which cannot exist in the table (0.5 point)
      - If they exist already, the new vendor cannot be inserted into the table and the system prints the information "The vendor id has already existed in the database" or "The vendor name has already existed in the database" (0.25 point)
      - If they do not exist, the new vendor is inserted into the table (0.5 point)
   b. (0.5 point) After creating procedure ***p_fudgemart_add_new_vendor in (a)***, write the statements to execute it for inserting vendors as below:

Vendor_id	Vendor_name	Vendor_phone	Vendor_website
21	'Vinmart'	555-2222	www.vinmart.com
22	'Mikerosoft'	555-8888	www.microsoft.com

   c. (1.25 point) Write a stored procedure (named as ***p_fudgemart_update_price_product***) which is used to update the whole sale price or the retail price of the product in the table fudgemart_products with the below requirement:

- Input: @product_id, @diff_price, @mode

  Description:

  If @diff_price is smaller or equal than 0, the stored procedure will print the information "the diff price must be greater than 0" and then return 0. (0.25 point). Otherwise, update the price of the product with its id equals @product_id and

  If @mode equals 1, the retail price is changed into the current whole sale price + @diff_price (0.5 point)

  If @mode equals 0, the whole sale price is changed into the current retail price - @diff_price (0.5 point)

d. (1 point) Write a trigger which is used to check the below things before inserting a new agent into the table fudgemart_agents :

   Check the country value:
   - If null, insert the new agent with the country code 'VN'. (0.25 point)
   - If not null, check whether it exists in the table country or not. If not exist, create this new country code in the table countries. (0.25 point)

   - The agent_created_date value is not necessary to be inserted. If it is Null, the procedure automatically gets the current date (at the time of adding data) to insert it for a new agent. (0.25 point)
   - If everything is ok, insert the new agent into the table. (0.25 point)

6. Functions (3 point)

   a. (1.5 point) Write a function (named as ***f_fudgemart_vendor_product_count***) as the below the requirement
   - Input: @vendor_id
   - Output: Return the number of active products that vendor @vendor_id provides. If it is greater than 0, print these product information group by product department according to the below format:

```
Product department: ….
        Product id – product name
        Product id – product name
Product department: ….
        Product id – product name
        Product id – product name
….
```

   b. (0.5 point) After creating the function ***f_fudgemart_vendor_product_count***, write a SELECT statement to display the list of vendors including vendor_id, vendor_name, number of active products in which number of active products are calculated from calling the function ***f_fudgemart_vendor_product_count***.

   c. (1 point) Write a function (named as ***f_fudgemart_employees_sum_salary***) with the below requirement
   - Input: @from_date, @to_date
   - Output: the total money that is paid for employees from the date @from_date to the date @to_date. The amount is calculated by summing of employees_hourlywage multiply with the number of hours of each employee working from the date @from_date to the date @to_date.