CEN 202 - Database Project **Ledia Leka**

Cosmetics Store Database

1. Problem Description

a) Introduction of the problem (Problem Analysis)

The business I have chosen to create a database for is a cosmetic chain of personal care and beauty stores. These stores offer beauty products including cosmetics, skincare, body, fragrance, beauty tools and haircare from numerous brands from all over the world. This cosmetic store will have its own online page where people can buy online, will have a member-only card which will provide loyal clients with a program which will offer them exclusive benefits and rewards. The products can be bought by using cash, credit cards and debit cards. This company will have numerous employees for each sector, managers for the employees and the stores and even marketing members. These few marketing members manage the marketing for different departments, one or more than one. Marketing can be done by different ways, for example by using models, by social media influencers, or ads in different platforms or even billboards, etc. Different products for a certain amount of time can have discounts. These discounts are percentages to lower the price so the clients can buy them cheaper. Tax is applied on price, tax is another percentage which is added on the price of the product.

b) Full description of the requirements taken into consideration

A database application that stores data about employees, departments, products, brands, membership clients and invoices.

The requirements are:

1. For each **employee** to have its employee_id, name, age, address, phone number, email, salary, work position and the department_id where he/she works stored.
2. For each **department** add the department_id, department name and the manager who runs it to the entity.
3. For each **product** store its id, name, brand, expire and produce date, description, quantity, price, the brands name who has produced it and the department where it is sold.
4. We should have the **brand** with its name stored, the names of brand are unique and they are identified by it. Also there should be a contact form to the band listed and product_ids from that brand currently at the store.
5. **Membership clients** should have their name, address, contact form, points, registration date stored. They will be identified by their client_id. Their birthday should be listed as well, for gifts or wishes.
6. An **invoice** should include the total balance, way of payment(cash or card), date, tax, client_id if it is a member, employee_id of the cashier, the id of each product and its own id.
7. A **discount** table should include the product_id which got the discount, date when it starts, its duration and what is the percentage of discount.
8. A **marketing** table will be completed with the id of the department where it runs, the id of the employee that manages it, the cost of the marketing campaign, the platform where it will run, which may be many, and the number of people it will reach in the area.
9. An employee which works on a certain department prepares the invoice.

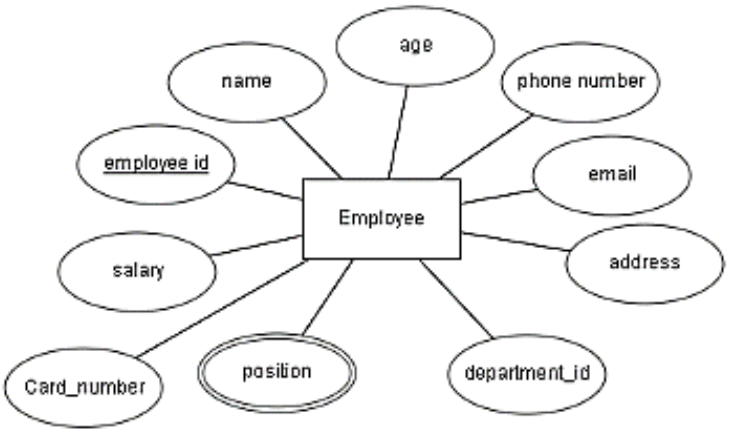
10. The brands produce the products which then are included in the invoice.
11. Client members are also included in the invoice to add their points for later gifts.
12. Each product can only be part of a certain department and only have one discount.
13. Each department will have a marketing campaign, which will be directed by a marketing employee.

2. Analysis with ER Diagrams

a) **Main Entities, Relationships and Cardinalities.**
Entity is a unique and distinct object used to collect and store data. Every entity has its own attributes which are characteristics of an entity.

Main entities:

Employee, department, products, membership clients, brands, invoice, discounts, marketing.



Employee:

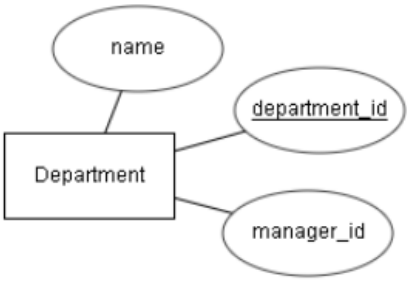
Employee_id – primary key

Position – multivalued key

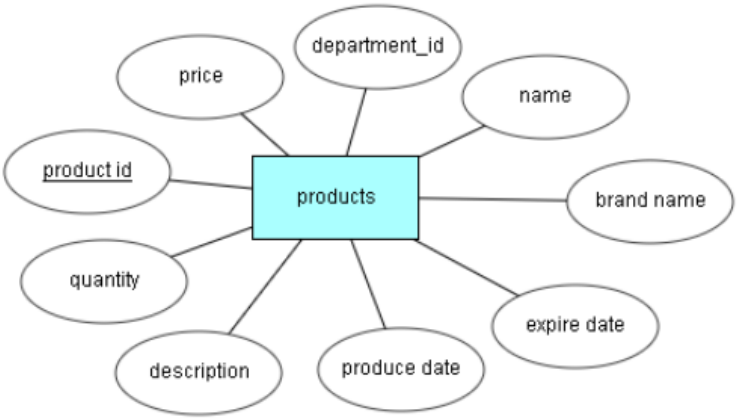
Department_id – foreign key

Department:

Department_id – primary key , unique



Products:



Product_id – primary key

Department_id – foreign key

Brand_name – foreign key

Invoice:

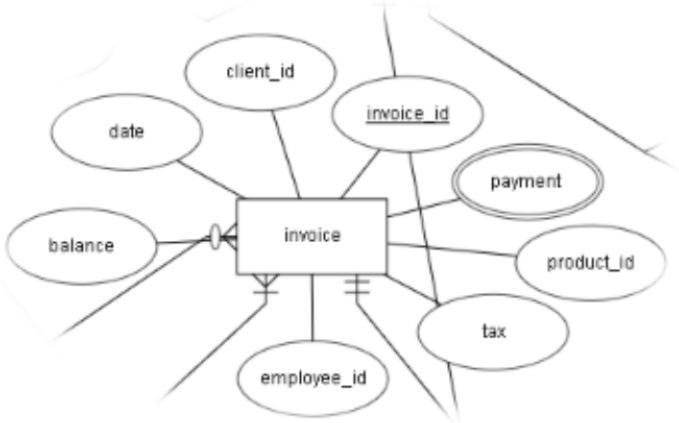
invoice_id – primary key

employee_id – foreign key

product_id – foreign key

client_id – foreign key

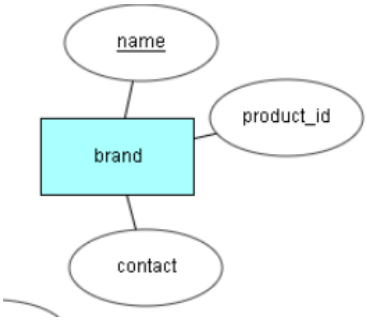
payment – multivalued key



Brand:

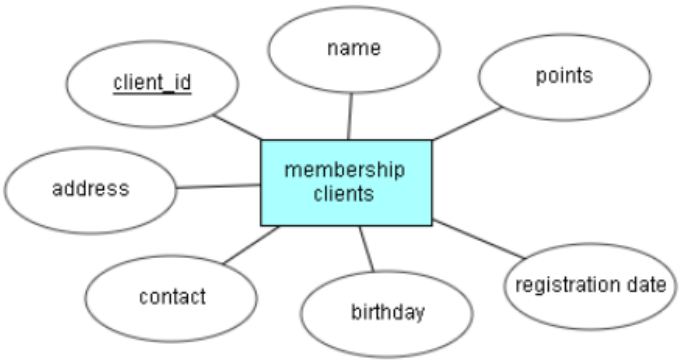
brand_name – primary key, unique

product_id – foreign key



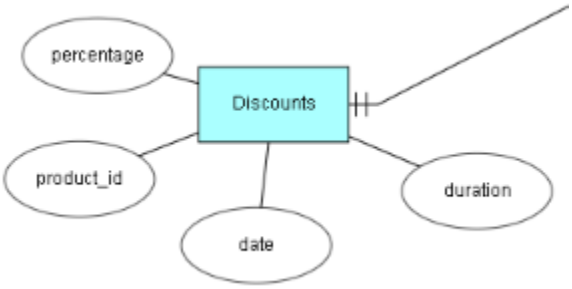
Member client:

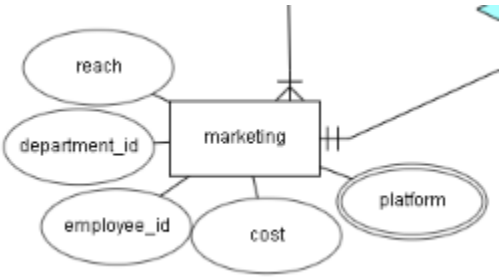
client_id – primary key



Discounts:

Product_id- foreign key





<u>Marketing:</u>

Department_id – foreign key

Employee_id – foreign key

Platform - multivalued

Relationships and their cardinalities:

A relationship describes an association among entities and its cardinalities are types of connections between ty entities, for example 1 to Many, 1 to 1, M to 1.

Works_on: employee_department: 1:1, 1 employee works on 1 department

Orders: membership client_invoice: 1:M,1 client orders many invoices

Includes: invoice_products: 1:M, 1 invoice has many products

Created_by: product_brands: 1:1, 1 product is created by 1 brand

Prepares: employee_invoice: 1:M, 1 employee prepares many invoices

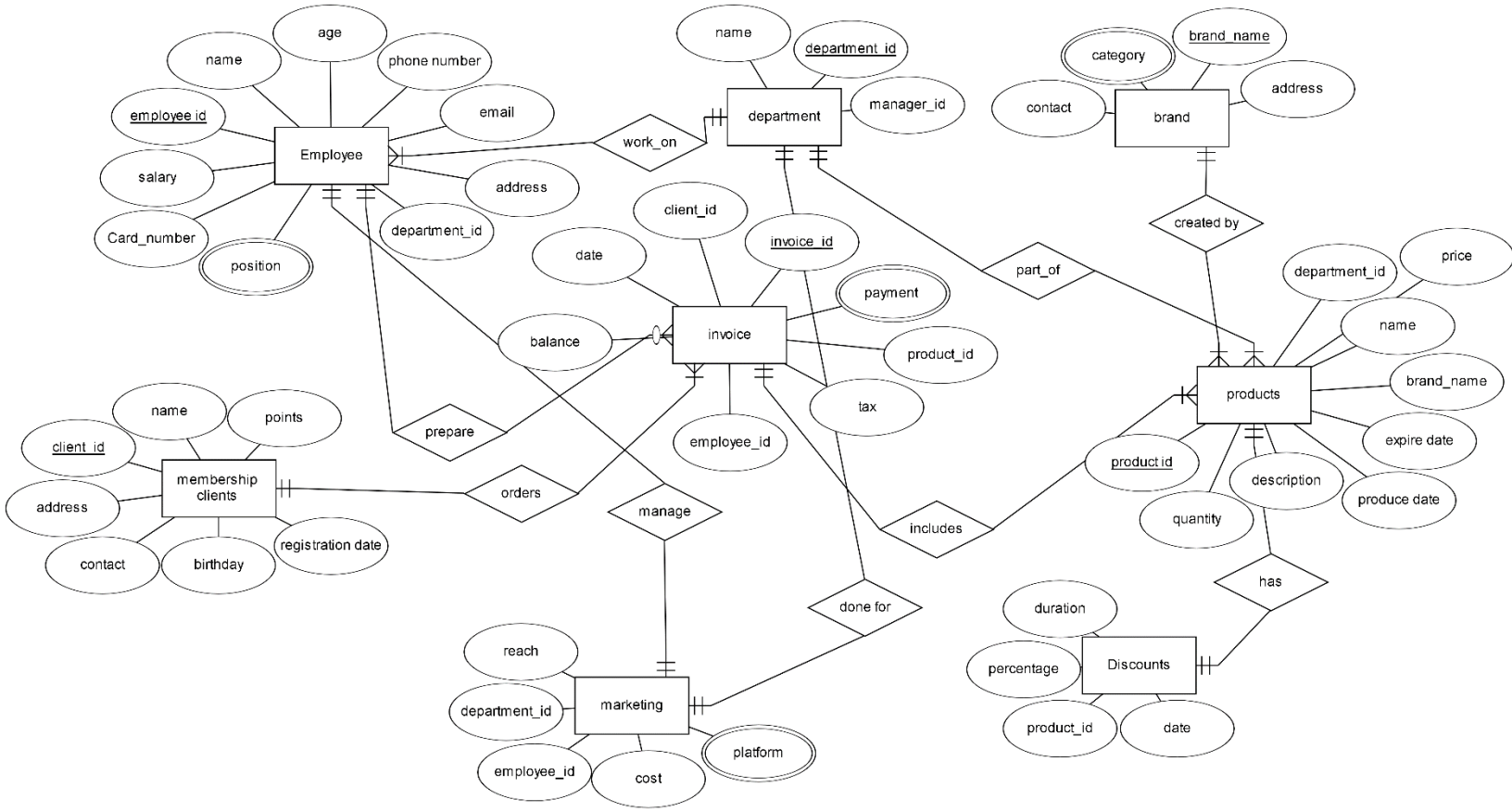
Part_of: product_department: 1:1, 1 product is part of 1 department

Has: products_discounts:1:1,1 product has 1 discount

Manages: employee_marketing: 1:1, 1 employee manages 1 marketing campaign

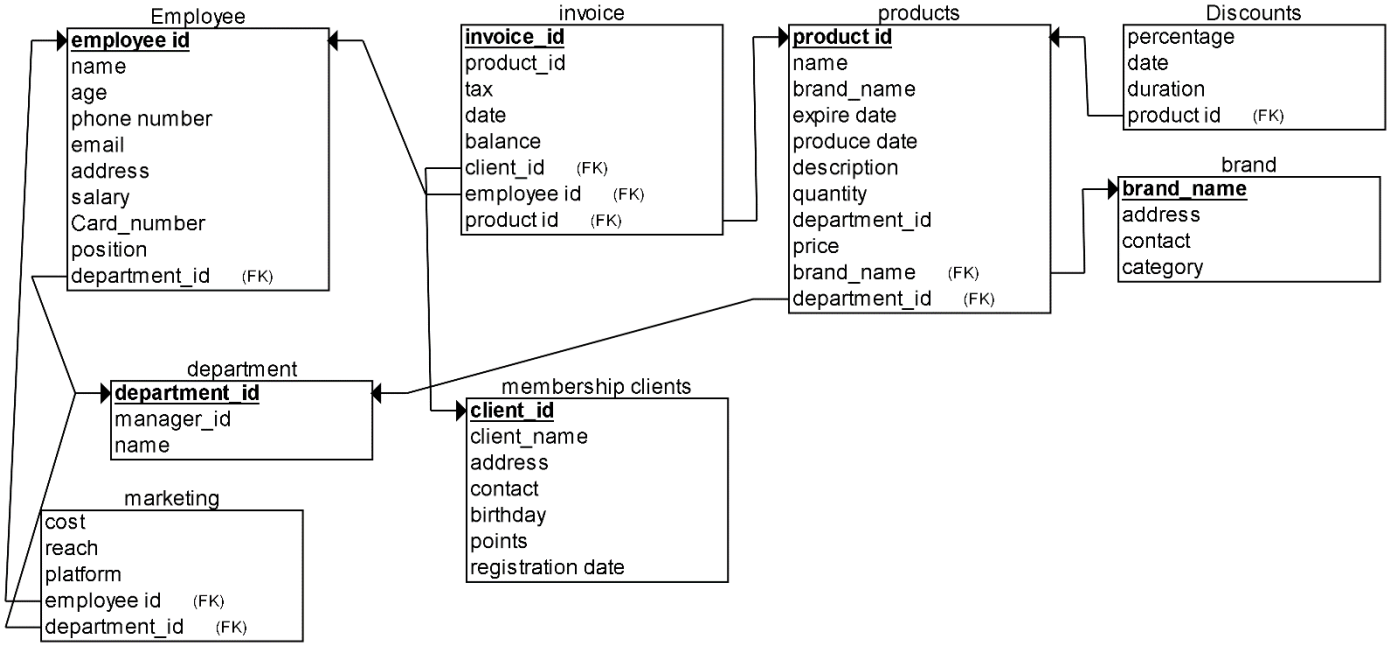
Done_for: marketing_department:1:1, 1 marketing campaign for 1 department

b)Full ER diagram



3. Relational Schema Model

a) The converted Relational Schema from ER Diagram



4. Rationale

a) Constraints, Triggers, Functions and Procedures

Constraints are used in Sql to specify rules for the data in a table. They limit the type of data that can go into a table, so if there is any violation of the constraint, the action is aborted. Constraints can apply to a column or a table. Some of the most used constraints are:

Not null – you can not leave a column value empty/NULL

Unique – all values is the column are different

Primary key – identifies all rows in the table, is unique and cannot be null

Foreign key – identifies rows in another table

Etc.

From the Relation Schema we can see that in my database most of the tables have a primary key. Employee_id is the primary key of the employee table, it identifies each employee in the store, this key is also a foreign key in the marketing table, which identifies the manager of marketing, is a foreign key in the invoice table, which identifies the cashier who has released the invoice. Department_id is another primary key which identifies each department and is used as a foreign key in the marketing table to relate each campaign with its department, its a foreign key in the products table to show where they belong, in the discounts table, to show the products which got a discount and in the invoice to show the product that was sold.

Invoice_id is a primary key which identifies invoices from one another, also acting as a series number. Client_id as well is used in the membership clients table to identify the clients which are members and then this id is used in the invoice when this member buys a product. Brand_name is used as identifier for the brands which produce the products

and it cannot be the same with another name, also it works as a foreign key for the products.

Triggers are the SQL codes that are automatically executed in response to certain events on a particular table, insert, update or delete. These are used to maintain the integrity of data.
Triggers can be before and after insert, update or delete keys. They execute automatically inside the database so the user cannot really notice its function. By creating a trigger you specify the trigger name, action time and action, the name of the table that it affects and the trigger body. There are three triggers in my code, first trigger is a before update trigger which displays an error message if the updated address of an employee is outside of the city where the store is. This error message makes it impossible to enter an unwanted address and does not make itself visible when you use the database.
The second trigger is an after insert trigger which automatically invokes itself after an insert event occurs in the table. So basically, after inserting a row in the table, the trigger is going to check the quantity of the products and if it has less than the predetermined value, another number will be added.
The last trigger is a before update trigger which validates data before it is updated to the table. This trigger in my case basically does not allow you to ender a null value as an address or as a contact when you update a value at the brand table.

A stored function is a special kind stored program that returns a single value. Typically, you use stored functions to encapsulate common formulas or business rules that are reusable among SQL statements or stored programs.
To create a stored function we use the create function statement, give it a name and then list all the parameters inside the parenthesis. Then we specify the return statement and add the key not/ deterministic then we write the body of the function.
My first function is 'salaryupdate' and my parameter is salary as a float. This function will return a float, write the deterministic key and then the body of the function which is a if - else if which according to the salaries, increases them with different percentages and returns new salaries for the employees.
The next function is about the expire date of the products. 'Expiresale' gets as parameters the expire_date and the sale and returns a new price for the products that are near the expire date.
The body of the function checks if the expire date is of this year's and if it is true, it decreases the price of the product by 50% and then returns the new price.
Another function is about getting the last price of a product. This last price is about the products that get a discount and also a tax raise. Its parameters are the tax, percentage and price floats and it will return another price float. The body will be a mathematical expression that calculates the new price. Then the function returns it. These functions can be used in queries and even replace the old price columns.

A stored procedure is a segment of declarative SQL statements stored inside the MySQL Server. If you want to save a query on the database server for execution later, one way to do it is to use a stored procedure. Once you save the stored procedure, you can invoke it by using the CALL statement and the statement with give the same result as running the

query. A stored procedure can have parameters so we can pass values to it and get back the results. Inside the body of a procedure can be used if conditions, loops , case, etc.
To create a procedure is enough to write that same statement and give it a name, if we want to pass parameters, write into if it an input and an output, or in /out name of parameter and its datatype otherwise. Then we write the bode of a procedure, nearly the same as the function.
In my code the first procedure is about giving a gift or not to a customer.
Returngift() gets as parameter 'out result char' which is a statement that gets returned if the points selected from membership_clients are greater than certain value to get the gift, or not to not get it. This is done by using if-else conditions and returning the result statement.
The other procedure is about counting employees from a certain city and getting their work position. This is done by giving as input to the procedure the city name as a char and the body is made by a number of select statements which compare the inputted city with the employee address and get their work position, adding employee number to the count which it later returns.
The next procedure is about running out of the products, it gets as input the product_id, and inout the count(used as input and output at the same time). So if product id is the same with that of a product, the counter decreases by lowering the quantity of products.
Another procedure is about expire date. Its parameter is an output date. The body of the procedure checks if the expire date of a product is of this years and then changes it to a few years later. This procedure returns the new expire date.

b. Authorization (User Access Definitions)

When the SQL standard authorization mode is enabled, object owners can use the GRANT and REVOKE SQL statements to set the user privileges for specific database objects or for specific SQL actions. They can also use roles to administer privileges. The GRANT statement is used to grant specific privileges to users or to roles, or to grant roles to users or to roles. The REVOKE statement is used to revoke privileges and role grants. The grant and revoke privileges are: delete, execute, insert, select, references, trigger, update. When a table, view, function, or procedure is created, the person that creates the object is referred to as the owner of the object. Only the object owner and the database owner have full privileges on the object. No other users have privileges on the object until the object owner grants privileges to them. Another way of saying that privileges on objects belong to the owner is to call them definer rights, as opposed to invoker rights. This is the terminology used by the SQL standard. SQL provides create, and drop operations for creating and deleting user accounts, and grant and revoke operations for managing privileges. A new user account is created with the statement 'create user'. A role is created with the statement 'create role': We may grant privileges to roles. We may assign roles to users. We may assign also privileges directly to user. We may also change users by using alter and rename. We may drop also users, revoke privileges from user, removing roles, revoke privileges from roles/