UPPSALA
UNIVERSITET

Project in Database Design I (1DL301)
AltOnline database
Group 21

Daniel Hjelm
Daniel.Hjelm.0958@student.uu.se

Emanuel Wreeby
Emanuel.Wreeby.1646@student.uu.se

Jakob Häggström
Jakob.Haggstrom.0348@student.uu.se

Urlich Icimpaye
Urlich.Icimpaye.1560@student.uu.se

Oscar Jacobson
Oscar.Jacobson.9201@student.uu.se

December 29, 2022

1 Milestone 1

1.1 Assumptions

Assumptions made:

- The store logo is assumed to be stored in the front-end application, hence must not be stored in the database.
- Breadcrumbs are constructed from the department title.
- The welcome text is the description of the root department.
- The add to basket button must not be stored since it is easy to just disable it when the stock quantity is zero.
- Average rating of a product is easily derived from the mean value of the number of stars a product has and therefore must not be stored in the database.
- A password should not be stored in a database for security reasons, hence a hashed version of it is stored instead.
- We accept one REVIEW from each USER for a specific PRODUCT.

1.2 ER-diagram

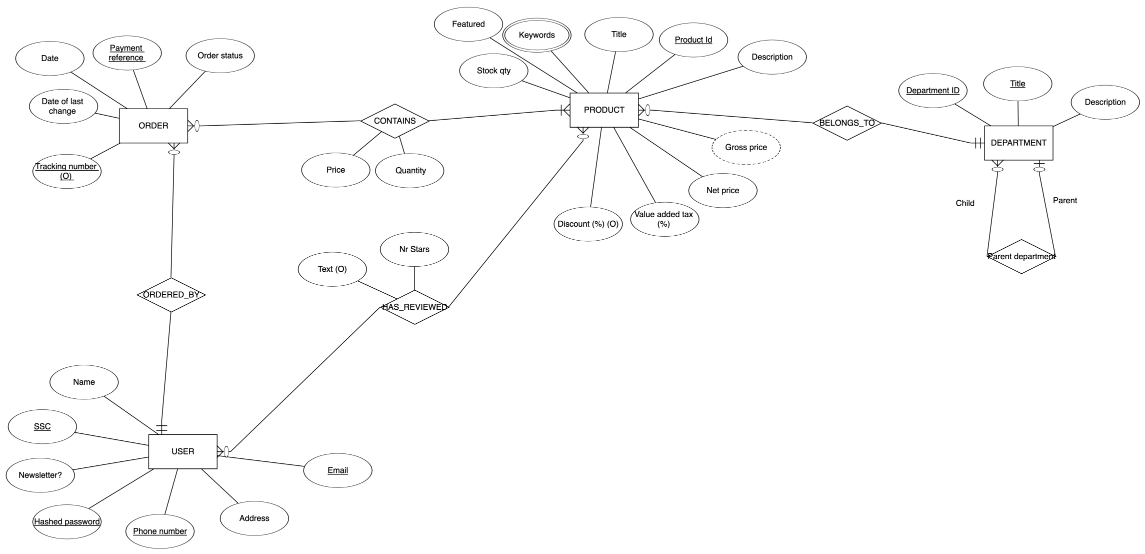

Figure 1: ER-diagram for the project database.

2 Milestone 2

2.1 Normalization

2.1.1 Unnormalized form (UNF)

In the figure below the unnormalized form, i.e. directly translated from the ER-diagram to a relational model, of the project database is displayed:

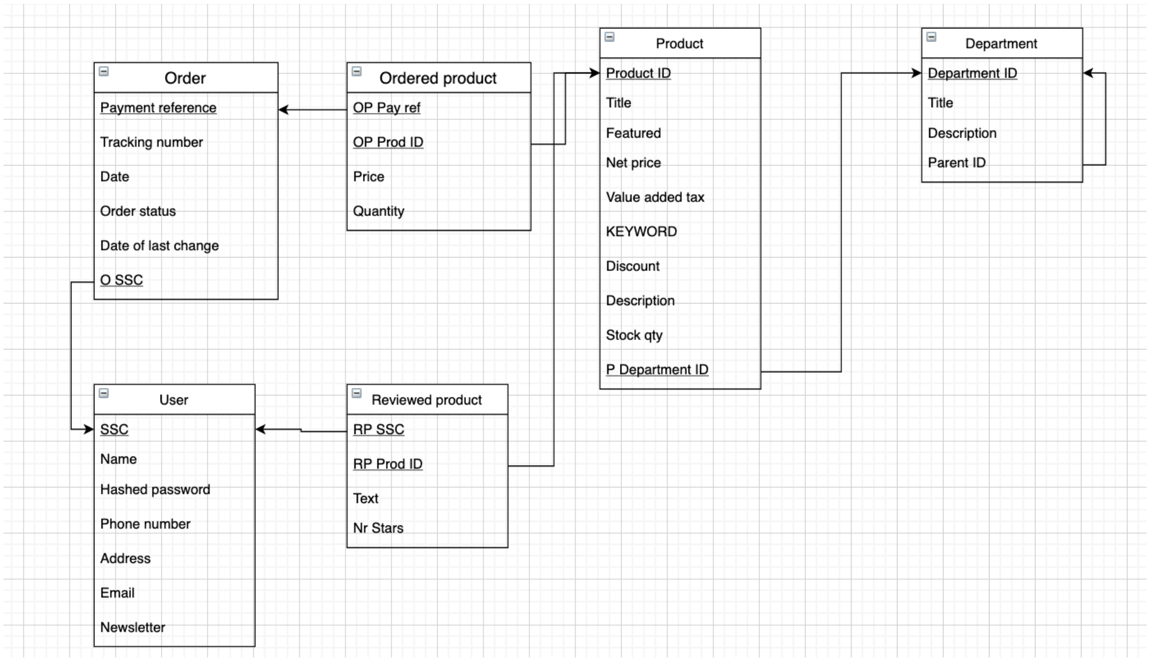

Figure 2: UNF of the project database.

2.1.2 Normalized form (1NF, 2NF, 3NF)

In the 1NF, composite and multivalued attributes are disallowed. Since Keywords in the PRODUCT relation is a multivalued attribute we have to move it to another table with a foreign key to the Product ID and a key representing the keyword.

In the 2NF, all attributes should depend on the whole key. KEYWORD, ORDERED PRODUCT and REVIEWED PRODUCT are the only relations with several keys which means we have to investigate them further. KEYWORD consist only of a composite key, hence depend on the whole key. In the ORDERED PRODUCT relation, both attributes price and quantity depend on the whole key since the price and quantity can differ for different products and orders. The same goes for REVIEWED PRODUCT, the text and number of stars can differ for different users and products.

In the 3NF, all attributes should depend on nothing but the key. As the relations are looking right now, no attributes depend on each other. Consequently, we have achieved the third normal form. The normalized model of the database is displayed in the figure below:

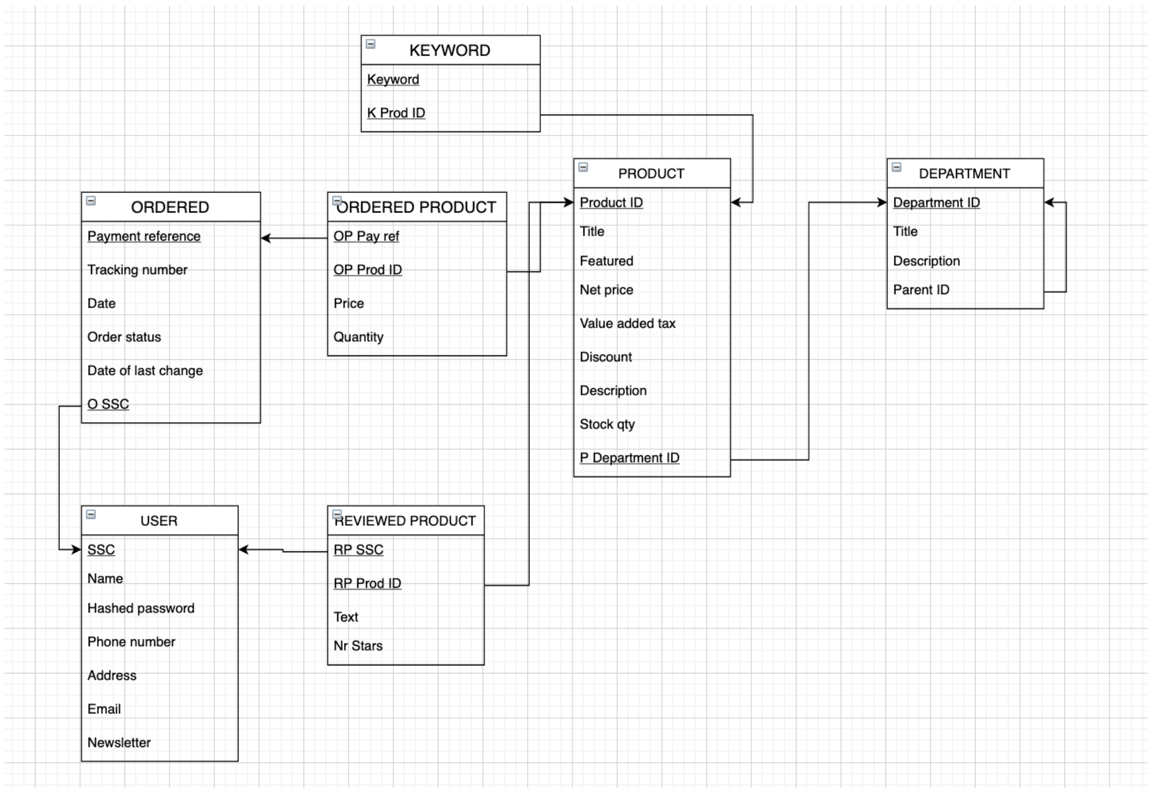

Figure 3: 1NF, 2NF and 3NF of the project database.

3 Milestone 3

3.1 SQL: Generate and populate database

The code for generating and populating can be found in the in sections 4.1.1 and 4.1.2 respectively.

3.2 MySQL Workbench's Reverse Engineer diagram

The diagram generated by the MySQL Workbench's Reverse Engineer functionality is presented in the figure below:

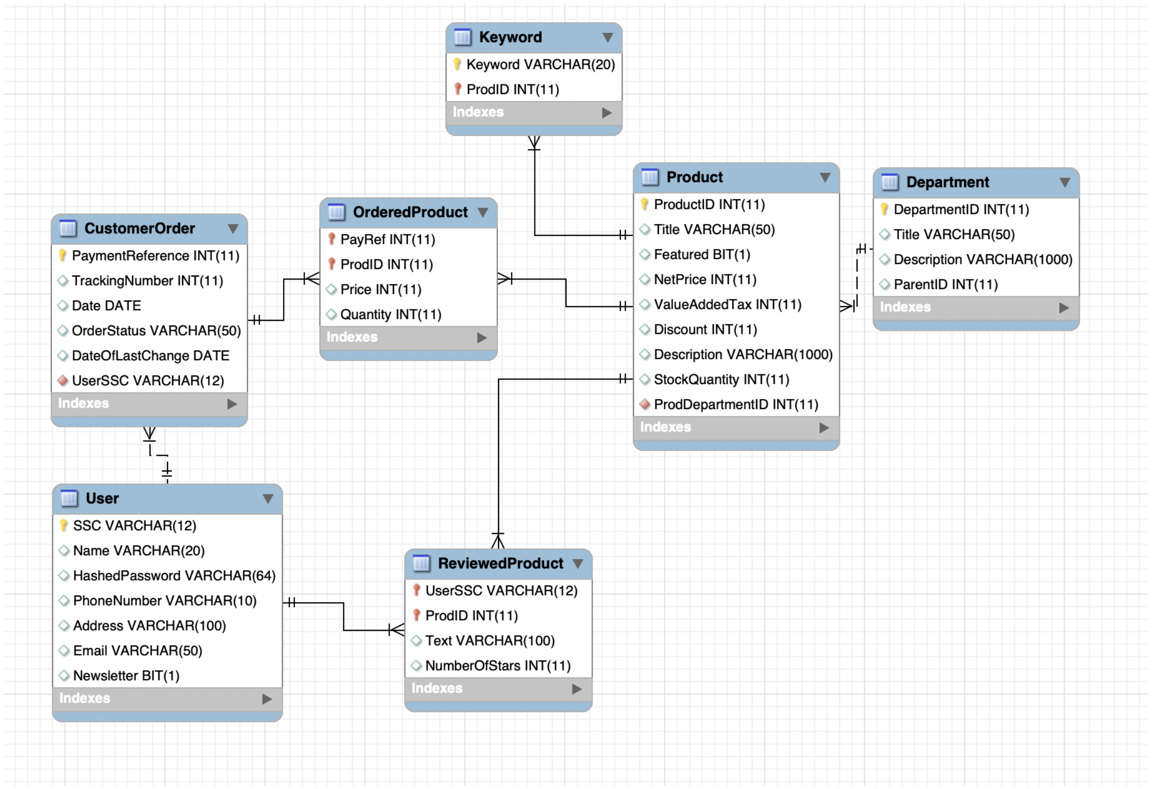

Figure 4: Diagram generated by MySQL Workbench's Reverse Engineer

3.3 SQL: Queries

The code for the queries can be found in Section 4.1.3.

3.4 SQL: Create indices

The code for the indices can be found in Section 4.1.4. Without using an index, we have the following table for the querying a list of given department all products with their average rating :

Table	Rows
Product	3
ReviewedProduct	4
Department	9

but when using an index for the department title it results in:

Table	Rows
Product	3
ReviewedProduct	4
Department	1

As a result, we have lessened the amount of rows that the database examines by 8. As for the query for finding the keyword-related products for a given product, we have the following table without using any index:

Tables	Rows
Keyword	3
Keyword	2
Product	1
Product	10

but when using an index for product title it results in:

Tables	Rows
Keyword	3
Keyword	2
Product	1
Product	1

As a result, we have lessened the amount of rows that the database examines by 9.

4 Milestone 4: Full version

4.1 Source code

The source code for generating, populating and querying the database together with the code to create indices and code to a program written in Python can be found in the subsections below.

4.1.1 Generate database

```
/* Assignment 4: Populate database */
CREATE TABLE Department (
   DepartmentID INT NOT NULL,
   Title VARCHAR(50),
   Description VARCHAR(1000),
   ParentID INT REFERENCES Department(DepartmentID),
  PRIMARY KEY (DepartmentID)
);
CREATE Table Product (
   ProductID INT NOT NULL,
   Title VARCHAR(50),
   Featured BIT,
   /* Boolean */
   NetPrice INT,
   ValueAddedTax INT,
   Discount INT,
   Description VARCHAR(1000),
   StockQuantity INT,
  ProdDepartmentID INT NOT NULL,
  PRIMARY Key (ProductID),
  FOREIGN KEY (ProdDepartmentID) REFERENCES Department(DepartmentID)
);
CREATE Table Keyword (
  Keyword VARCHAR(20),
  ProdID INT NOT NULL,
  FOREIGN KEY (ProdID) REFERENCES Product(ProductID),
  PRIMARY KEY (Keyword, ProdID)
  /* Composite key */
);
CREATE Table User (
  SSC VARCHAR(12) NOT NULL,
  Name VARCHAR(20),
  HashedPassword VARCHAR(64),
  PhoneNumber VARCHAR(10),
  Address VARCHAR(100),
  Email VARCHAR(50),
  Newsletter BIT,
  /* Boolean */
  PRIMARY KEY (SSC)
);
CREATE Table CustomerOrder (
  PaymentReference INT NOT NULL,
  TrackingNumber INT UNIQUE,
  Date DATE,
  OrderStatus VARCHAR(50),
  DateOfLastChange DATE,
  UserSSC VARCHAR(12) NOT NULL,
  PRIMARY KEY (PaymentReference),
  FOREIGN KEY (UserSSC) REFERENCES User(SSC)
);
CREATE Table OrderedProduct (
  PayRef INT NOT NULL,
  ProdID INT NOT NULL,
  Price INT,
  Quantity INT,
  FOREIGN KEY (PayRef) REFERENCES CustomerOrder(PaymentReference),
  FOREIGN KEY (ProdID) REFERENCES Product(ProductID),
  PRIMARY KEY (PayRef, ProdID)
);
CREATE Table ReviewedProduct (
  UserSSC VARCHAR(12) NOT NULL,
  ProdID INT NOT NULL,
  Text VARCHAR(100),
  NumberOfStars INT,
  FOREIGN KEY (UserSSC) REFERENCES User(SSC),
  FOREIGN KEY (ProdID) REFERENCES Product(ProductID),
  PRIMARY KEY (UserSSC, ProdID),
  CONSTRAINT CHK_NumberOfStars CHECK (
    NumberOfStars >= 0
    AND NumberOfStars <= 5
  )
);
```

4.1.2 Populate database

```
/* Top-level departments */
INSERT INTO
  Department(DepartmentID, Title, ParentID, Description)
VALUES
  (
    1000,
    "Home",
    NULL,
    "Welcome to the AltOnline Homepage"
  ),
  (
    1100,
    "TV and Video",
    1000,
    "Here you find the TV and Video stuff"
  ),
  (
    1200,
    "Computers and Tablets",
    1000,
    "Here you find the computers and tablets"
  );
  /* 3 child departments */
INSERT INTO
  Department(DepartmentID, Title, ParentID, Description)
VALUES
  (
    1110,
    "TV",
    1100,
    "Here you find the TV stuff"
  ),
  (
    1120,
    "Streaming",
    1100,
    "Here you find the streaming services"
  ),
  (
    1130,
    "Blu-Ray and DVD",
    1100,
    "Here you find the Blu-ray and DVD:s"
  ),
  (
    1210,
    "Computers",
    1200,
    "Welcome to the computers"
  ),
  (
    1220,
    "Laptops",
    1200,
    "Here you find the laptops"
  ),
  (
    1230,
    "Tablets",
    1200,
    "Here you find the tablets"
  );
  /* Products */
INSERT INTO
  Product(
    ProductID,
    Title,
    Featured,
    NetPrice,
    ValueAddedTax,
    Discount,
    Description,
    StockQuantity,
    ProdDepartmentID
  )
VALUES(
    1,
    "Samsung big TV",
    1,
    1000,
    25,
    10,
    "Big␣TV",
    10,
    1110
),
(
    2,
    "LG␣big␣TV",
    1,
    1200,
    25,
    50,
    "Bigger␣TV",
    5,
    1110
),
(
    3,
    "Nokia␣big␣TV",
    1,
    2000,
    25,
    0,
    "Biggest␣TV",
    2,
    1110
),
(
    4,
    "Macbook␣Air",
    0,
    1000,
    25,
    0,
    "Nice␣computer",
    5,
    1220
),
(
    5,
    "Macbook␣Pro",
    0,
    2000,
    25,
    0,
    "Nice␣computer",
    2,
    1220
),
(
    6,
    "Macbook",
    0,
    800,
    25,
    20,
    "Nice␣computer",
    20,
    1220
),
(
    7,
    "Ipad␣Air",
    0,
    500,
    25,
    0,
    "Nice␣tablet",
    12,
    1230
),
(
    8,
    "Ipad␣Pro",
    0,
    800,
    25,
    0,
    "Nice␣tablet",
    5,
    1230
),
(
    9,
    "Ipad",
    0,
    200,
    25,
    0,
    "Nice␣tablet",
    5,
    1230
),
  (
    10,
    "Best␣gaming␣computer␣EU",
    0,
    10000,
    25,
    0,
    "Look␣at␣the␣title␣duh",
    1,
    1210
  );
  /* Create 2 users */
INSERT INTO
  User (
    SSC,
    Name,
    HashedPassword,
    PhoneNumber,
    Address,
    Email,
    Newsletter
  )
VALUES
  (
    9804062454,
    "Daniel␣Hjelm",
    "213adasdeqr2e123#12312313",
    0793355974,
    "Luthagesesplanden␣81,␣Uppsala␣752␣71",
    "dnl1@live.se",
    1
  ),
  (
    9704319559,
    "Emanuel␣Wreeby",
    "ujlakfs293704hjalkfsd02ruo1r3",
    0732681670,
    "Studentv gen␣14,␣Uppsala␣752␣34",
    "manne.wreeby@gmail.com",
    0
  );
  /* Create 2 reviews */
INSERT INTO
  ReviewedProduct (
    UserSSC,
    ProdID,
    Text,
    NumberOfStars
  )
VALUES
  (
    9804062454,
    5,
    "Best␣computer␣on␣the␣market.␣Looking␣forward␣to␣testing␣out␣the␣new␣M1-chip",
    5
  ),
  (
    9704319559,
    5,
    "Almost␣a␣5/5",
    4
  ),
  (
    9704319559,
    3,
    "Almost␣a␣5/5",
    4
  ),
  (9804062454, 3, "Meh", 3);
INSERT INTO
  CustomerOrder (
    PaymentReference,
    TrackingNumber,
    Date,
    OrderStatus,
    DateOfLastChange,
    UserSSC
  )
VALUES
  (
    123456789,
    987654321,
    '2021-09-20',
    "Sent",
    '2021-09-20',
    9804062454
  );
INSERT INTO
  OrderedProduct (PayRef, ProdID, Price, Quantity)
VALUES
  (123456789, 5, 2000, 1);
INSERT INTO
```

```
  Keyword (Keyword, ProdID)
VALUES
  ("Apple", 4),
  ("Laptop", 4),
  ("New␣in", 4),
  ("Apple", 5),
  ("Laptop", 5),
  ("New␣in", 5),
  ("Apple", 6);
```

4.1.3 Queries

```
/* Assignment 5: SQL Queries */
  /* Welcome text for the homepage */
SELECT
  Description
FROM
  Department
WHERE
  ParentID IS NULL;
  /* Top-level departments */
SELECT
  Title,
  Description
FROM
  Department
WHERE
  ParentID = 1000;
  /* List of featured products */
SELECT
  Title,
  Description,
  NetPrice,
  ValueAddedTax,
  Discount,
  NetPrice * (1 + ValueAddedTax / 100) * (1 - Discount / 100) AS CurrentRetailPrice
FROM
  Product
WHERE
  Featured = true;
  /* Given a product, list all keyword-related products */
SELECT
  DISTINCT title,
  description
FROM
  Product P
  INNER JOIN Keyword K ON K.ProdID = P.ProductID
```

```
    AND K.Keyword IN (
        SELECT
            Keyword
        FROM
            Keyword
        WHERE
            ProdID = (
                SELECT
                    ProductID
                FROM
                    Product
                WHERE
                    title = "Macbook␣pro"
            )
    )
    AND P.title != "Macbook␣Pro";
    /* Given an department, list of all its products (title, short description, current r
            average rating */
SELECT
    Title,
    Description,
    NetPrice * (1 + ValueAddedTax / 100) * (1 - Discount / 100) AS CurrentRetailPrice,
    AVG(NumberOfStars) AS AverageRating
FROM
    Product
    JOIN ReviewedProduct ON Product.ProductID = ReviewedProduct.ProdID
    AND ProdDepartmentID = (
        SELECT
            DepartmentID
        FROM
            Department
        WHERE
            title = "Laptops"
    )
GROUP BY
    Product.ProductID;
    /* List of all products on sale sorted by the discount percentage (starting with the
SELECT
    *
FROM
    Product
WHERE
    Discount > 0
ORDER BY
    Discount DESC;
```

4.1.4 Indices

```
 /* Index */
CREATE INDEX DepartmentTitleOrder on Department(Title ASC);
CREATE INDEX ProdTitleOrder on Product(Title ASC);
EXPLAIN
SELECT
   Title,
   Description,
   NetPrice * (1 + ValueAddedTax / 100) * (1 - Discount / 100) AS CurrentRetailPrice,
  AVG(NumberOfStars) AS AverageRating
FROM
   Product
  JOIN ReviewedProduct ON Product.ProductID = ReviewedProduct.ProdID
  AND ProdDepartmentID = (
     SELECT
        DepartmentID
     FROM
        Department
     WHERE
        title = "Laptops"
   )
GROUP BY
   Product.ProductID;
  EXPLAIN SELECT
  DISTINCT title,
   description
FROM
   Product P
  INNER JOIN Keyword K ON K.ProdID = P.ProductID
  AND K.Keyword IN (
     SELECT
        Keyword
     FROM
        Keyword
     WHERE
        ProdID = (
          SELECT
             ProductID
          FROM
             Product
          WHERE
             title = "Macbook␣pro"
        )
   )
  AND P.title != "Macbook␣Pro";
```

```
DROP INDEX DepartmentTitleOrder on Department;
DROP INDEX ProdTitleOrder on Product;
```

4.1.5 Python program

```
import mysql.connector

def pythonProgram():
    '''Function that performs two different modes:
        1. Department mode:
        Ask the user for departmentID and lists all its products if the given department is
        2. Product mode:
        Asks for a product ID, shows the current discount and allows the user to change it.
    '''
    while True:
            modePick = input("Would␣you␣like␣run␣the␣department␣mode,␣please␣enter␣1.␣If␣you␣
            if modePick in ["1","2"]:
                break
            else:
                print("You␣did␣not␣enter␣a␣valid␣mode,␣please␣try␣again!")

    # Group number
    group_number="21"

    mydb = mysql.connector.connect(
        # host="groucho.it.uu.se",
        host = "127.0.0.1",
        user="ht21_1_group_"+group_number,
        passwd="pwd_"+group_number,
        database="ht21_1_project_group_"+group_number
    )

    # Create a cursor
    mycursor = mydb.cursor()

    # Department mode
    if modePick == "1":
        # Ask the user for department ID
        while True:
            # Fetch departmentID
            departmentID = input("Which␣department␣would␣you␣like␣to␣see␣(enter␣departmentID␣
            mycursor.execute("SELECT␣DepartmentID␣FROM␣Department␣WHERE␣DepartmentID␣=␣%s", (
            myresult = mycursor.fetchall()
            # If the departmentID exist, continue
            if (len(myresult)>0):
```

```
        break
      else:
        # Tell the user that the departmentID doesn't exist and let them try again.
        print("The departmentID does not exist in the database, please try again!")
        continue

  # Select products from the department
  mycursor.execute("SELECT ProductID, Title, NetPrice * (1 + ValueAddedTax / 100) * (
  myresult = mycursor.fetchall()
  # If the department has any products, list them
  if len(myresult) > 0:
    print("Products: ")
    # Convert to list to change the Netprice's type to int (from decimal)
    for product in myresult:
      output = []
      for i,element in enumerate(product):
        if i == 2:
          output.append(int(element))
        else:
          output.append(element)
      print(output)

  # Otherwise, list all department which has this department as its parent
  else:
    print("Child departments:")
    mycursor.execute("SELECT DepartmentID, Title FROM Department WHERE ParentID = %s"
    myresult = mycursor.fetchall()
    for department in myresult:
      print(department)

  mydb.close()

# Product mode
else:

  # Ask the user for productID
  while True:
    # Fetch productID
    productID = input("Which product would you like to see (enter productID please)?:
    mycursor.execute("SELECT Title, Discount from Product WHERE ProductID = %s", (pro
    myresult = mycursor.fetchall()
    # Check if it exists in database
    if (len(myresult)>0):
      break
    else:
      # Tell the user that the productID doesn't exist and let them try again.
```

```
        print("The productID does not exist in the database, please try again!")

    # Show the user the product
    for product in myresult:
      print(product)

    while True:
      # Ask if the user want to update the discount
      updateQuestion = input("Would you like to update the discount for this product? (
      # Check if they answer yes or no (y/n)
      if updateQuestion in ["y","n"]:
        break
      else:
        print("You did not enter a valid answer, please try again!")
    if updateQuestion == "y":
      while True:
        discount = int(input("Please enter wanted discount for the product: "))
        # Check that it is a valid discount
        if 0 <= discount <= 100:
          mycursor.execute("UPDATE Product SET Discount = %s WHERE ProductID = %s", (di
          mydb.commit()
          print(f'The new discount for product {productID} is {discount}')
          break
        else:
          print("You did not enter a valid discount , please try again!")



if __name__ == '__main__':
  pythonProgram()
```