Database Design I

Uppsala University

September 2024

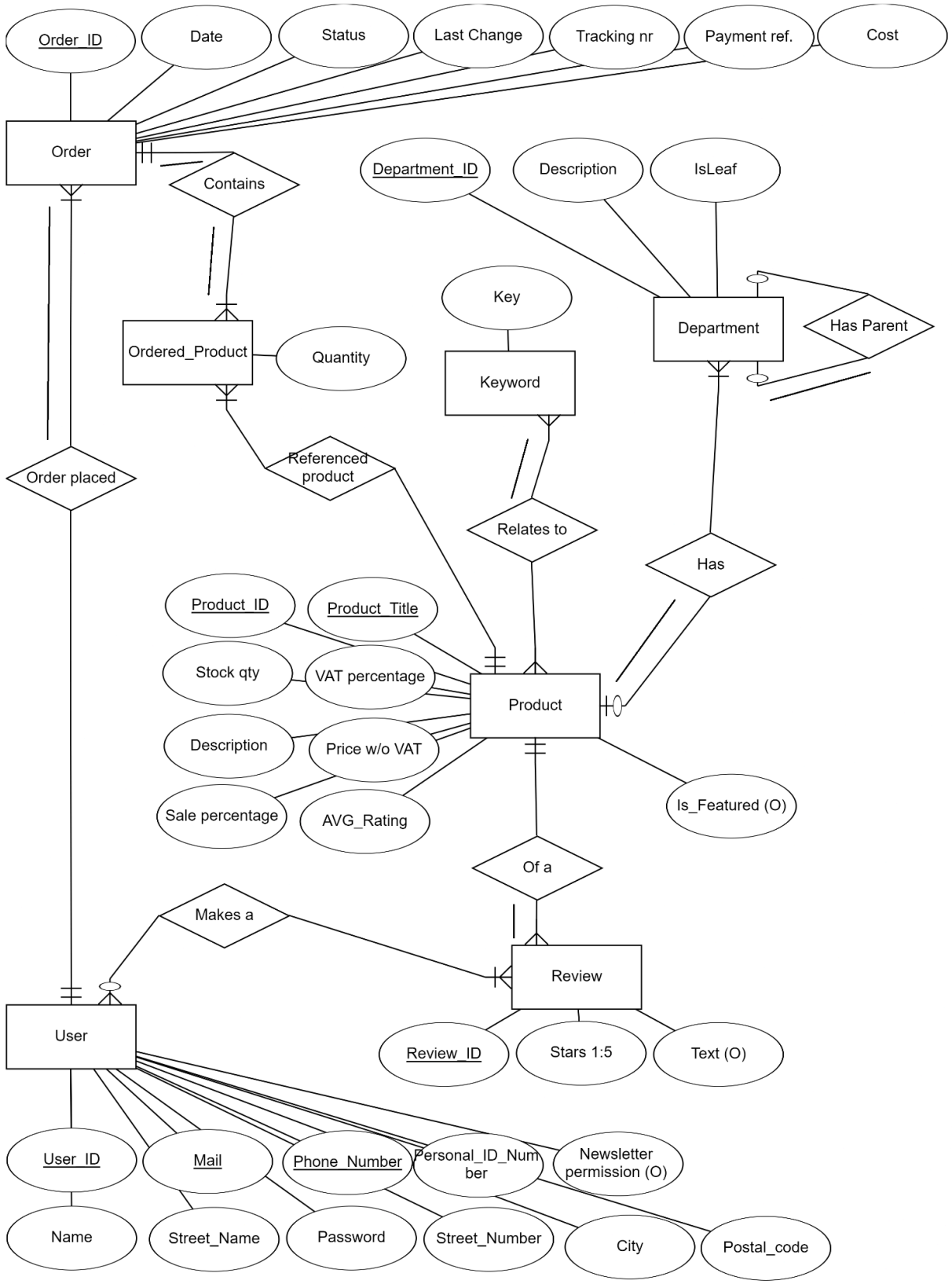
Milestone 1-4

Project Group 35:

Elias Olofsson: elias.olofsson.8606@student.uu.se
Mauritz Hamrin Sverredal: maha0549@student.uu.se
Ella Taawo: ella.taawo.3819@student.uu.se
Lovisa Hambäck: lovisa.hamback.7619@student.uu.se
Elias Swanberg: elias.swanberg.4884@student.uu.se

Task 1: EER Model

Image 1. EER model of AltOnline Database, updated after feedback



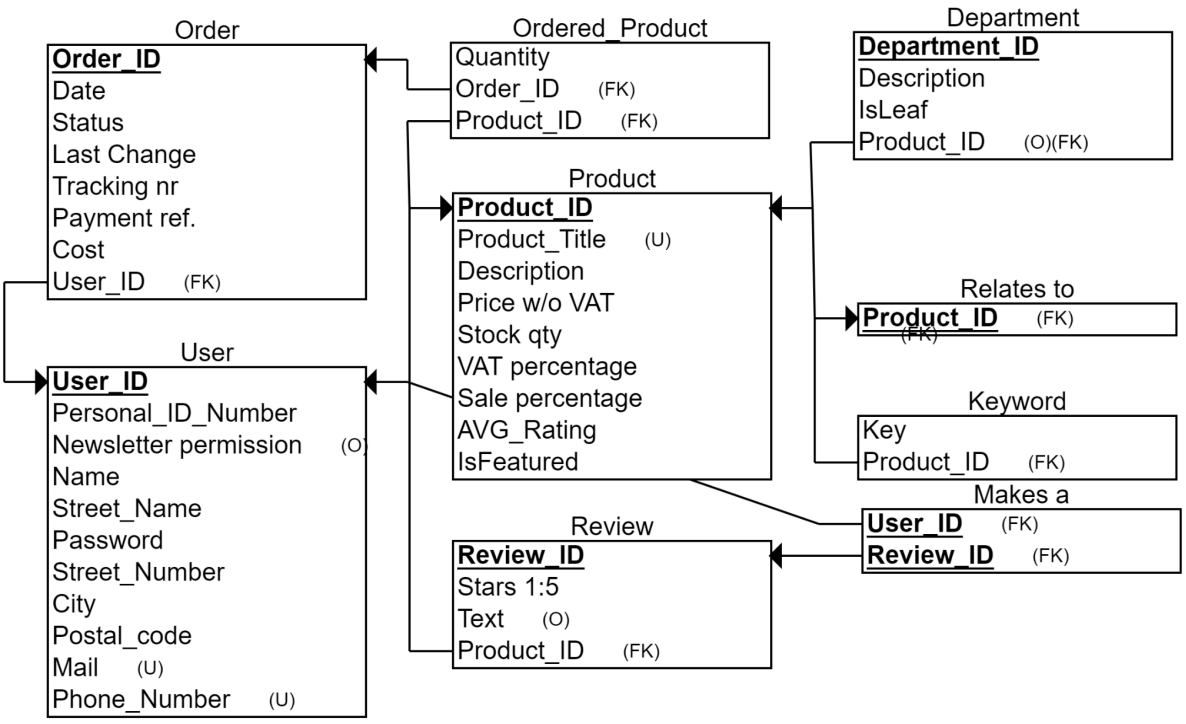
Total Participation is marked with double lines in the diagram.

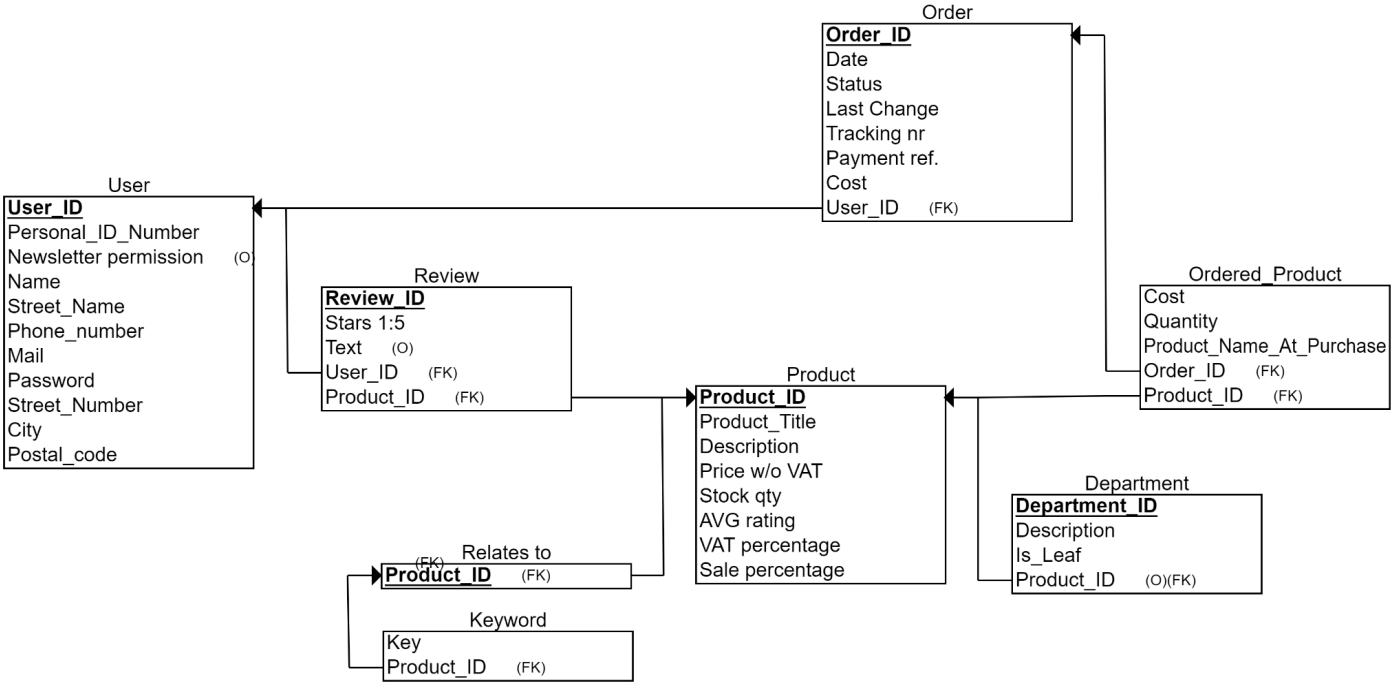
The assumptions we made when making the EER model were:

1. You don't need to make a review or place an order to be a user.
2. One user can make multiple orders and reviews, but they can only, and must, be connected to one user.
3. User_ID, personal_ID_Number and mail are unique for each user.
4. We created the entity ordered_product to store the product, costs and quantity when ordering separately, since the cost may be altered after the order was made.
5. Ordered_product must be connected to one user and one/multiple products.
6. AVG rating is calculated from the reviews associated to the product and is therefore not a stored attribute.
7. Price is calculated from price w/o VAT, VAT % and sale %.
8. Each product is identified using product_ID which is unique.
9. Sales are only to individuals and not businesses.
10. A product must belong to one department.
11. A department can be either Homepage and non-leaf, non-leaf, or leaf.
12. The homepage is simply a department with no parent department, and its welcome text is specified in the description attribute.
13. A non-leaf department can have other departments (both leaf and non-leaf) as children.
14. There can be two departments with the same name, therefore the name isn't unique. Instead, the unique identifier is the department_ID.
15. A department which has no children is a leaf department.
16. URL and Logo are not stored in the database.

Task 2: Normalization

Relational Schema:




Order
Order_ID
Date
Status
Last Change
Tracking nr
Payment ref.
Cost
User_ID (FK)
User
User_ID
Personal_ID_Number
Newsletter permission (O)
Name
Street_Name
Phone_number
Mail
Password
Street_Number
City
Postal_code
Review
Review_ID
Stars 1:5
Text (O)
User_ID (FK)
Product_ID (FK)
Ordered_Product
Cost
Quantity
Product_Name_At_Purchase
Order_ID (FK)
Product_ID (FK)
Product
Product_ID
Product_Title
Description
Price w/o VAT
Stock qty
AVG rating
VAT percentage
Sale percentage
Department
Department_ID
Description
Is_Leaf
Product_ID (O)(FK)
Relates to
(FK)
Product_ID (FK)
Keyword
Key
Product_ID (FK)

1NF Tables with example data:

(User)

User_ID (PK)	Personal_ID_Number (Unique)	Newsletter_Permission	Name	Phone_Number (Unique)	Mail (Unique)	Password	Street_Name	Street_Number	City	Postal_Code
1	19850821-1234	Yes	Alice Doe	701234567	alice@example.com	******	Main Street	12	Stockholm	11122
2	19901210-5678	No	Bob Smith	739876543	bob@example.com	******	Oak Avenue	34	Göteborg	41103
3	19750325-9876	Yes	Carol May	709988776	carol.may@mail.com	******	Pine Street	56	Malmö	21119
4	19830312-5432	Yes	David Lee	762233445	david.lee@provider.se	******	Cedar Lane	78	Uppsala	75175

(Order)

Order_ID (PK)	Date	Status	Last_Change	Tracking_nr (Unique)	Payment_ref (Unique)	Cost	User_ID (FK)
1000	2000-01-01	Shipped	2000-03-01	TRK00314	PAY01234	999	8
1001	2024-09-10	Shipped	2024-09-11	TRK12345	PAY54321	250	1
1002	2024-09-15	Processing	2024-09-16	TRK67890	PAY98765	150	2
1003	2024-09-20	Delivered	2024-09-21	TRK13579	PAY24680	300	3

(Review) Primary key (User_ID, Product_ID)

User_ID (FK)	Product_ID (FK)	Stars	Text
1	3001	5	Excellent product!
2	3002	3	It was okay.
3	3003	4	Good value for money

(Product)

Product_ID (PK)	Product_Title	Description	Price_excl_VAT	Stock_Qty	VAT_category (FK)	Sale_percentage	Is_Featured	Department_ID (FK)
3001	Smartphone	Latest model smartphone	500	10	1	10	True	4001
3002	Laptop	High-performance laptop	1200	5	1	15	False	4002
3003	NC_Headphones	Noise-cancelling headphones	150	20	1	5	False	4003
3004	WL_Headphones	Wireless headphones	150	20	1	5	True	4003

(VAT_Category)

ID (PK)	VAT percentage
1	25%
2	12%
3	6%
4	0%

(Ordered_Product)

Ordered_product_ID (PK)	Cost	Quantity	Product_Name_At_Purchase	Order_ID (FK)
1	250	1	Smartphone	1001
2	150	2	Laptop	1002
3	999	5	Rotary phone	1000

4	300	3	Headphones	1003

(Referenced_Product)

Ordered_Product_ID (PK) (FK)	Product_ID (FK)
1	3001
2	3002
4	3003

(Department)

Department_ID (PK)	Name	Description	Parent_Departement (FK)
4000	Homepage	Welcome text	4000
4001	Electronics	…	4000
4002	Computers	PCs and laptops	4001
4003	Accessories	Accessories for computers	4002

(Keyword)

ID (PK)	Name
1	Mobile
2	Computer
3	Audio
4	Wireless

(Keyword_Assignments)

Keyword_ID (FK)	Product_ID (FK)
1	3001
2	3002
3	3003
3	3004
4	3004

2NF:

Changes made to comply with 2NF - All non-key attributes are fully dependent on the key

No changes

3NF:

Separated Phonenumbers and Emails to avoid transitive dependencies (non-key attributes which depend on other non-key attributes). Separated payment ref and tracking number to avoid transitive dependencies.
These are the relations which we modified in order to comply with 3NF rules.

(User)

User_ID (PK)	Newsletter_Permission	Name	Phone_Number (Unique)	Mail (Unique)	Password	Street_Name	Street_Number	City	Postal_Code
1	Yes	Alice Doe	701234567	alice@example.com	******	Main Street	12	Stockholm	11122
2	No	Bob Smith	739876543	bob@example.com	******	Oak Avenue	34	Göteborg	41103
3	Yes	Carol May	709988776	carol.may@mail.com	******	Pine Street	56	Malmö	21119
4	Yes	David Lee	762233445	david.lee@provider.se	******	Cedar Lane	78	Uppsala	75175

(User_Phone_Number)

User_ID (PK) (FK)	Phone_Number (Unique)

1	701234567
2	739876543
3	709988776
4	762233445

(User_Email)

User_ID (PK) (FK)	Email_Address (Unique)
1	alice@example.com
2	bob@example.com
3	carol.may@mail.com
4	david.lee@provider.se

(User_PID)

User_ID (PK) (FK)	Personal_ID_Number (Unique)
1	19850821-1234
2	19901210-5678
3	19750325-9876
4	19830312-5432

(Order)

Order_ID (PK)	Date	Status	Last_Change	Cost	User_ID (FK)
1000	2000-01-01	Shipped	2000-03-01	999	8
1001	2024-09-10	Shipped	2024-09-11	250	1
1002	2024-09-15	Processing	2024-09-16	150	2
1003	2024-09-20	Delivered	2024-09-21	300	3

(Payment_ref)

Order_ID (PK)	Payment_ref (Unique)
1000	PAY01234
1001	PAY54321
1002	PAY98765
1003	PAY24680

(Tracking_nr)

Order_ID (PK)	Tracking_nr (Unique)
1000	TRK00314

1001	TRK12345
1002	TRK67890
1003	TRK13579

Complete 3NF tables:

(User)

User_ID (PK)	Newsletter_Permission	Name	Password	Street_Name	Street_Number	City	Postal_Code
1	Yes	Alice Doe	******	Main Street	12	Stockholm	11122
2	No	Bob Smith	******	Oak Avenue	34	Göteborg	41103
3	Yes	Carol May	******	Pine Street	56	Malmö	21119
4	Yes	David Lee	******	Cedar Lane	78	Uppsala	75175

(User_Phone_Numbers)

User_ID (PK) (FK)	Phone_Number (Unique)
1	701234567
2	739876543
3	709988776
4	762233445

(User_Emails)

User_ID (PK) (FK)	Email_Address (Unique)
1	alice@example.com
2	bob@example.com
3	carol.may@mail.com
4	david.lee@provider.se

(User_PID)

User_ID (PK) (FK)	Personal_ID_Number (Unique)
1	19850821-1234
2	19901210-5678
3	19750325-9876

4	19830312-5432

(Order)

Order_ID (PK)	Date	Status	Last_Change	Cost	User_ID (FK)
1000	2000-01-01	Shipped	2000-03-01	999	8
1001	2024-09-10	Shipped	2024-09-11	250	1
1002	2024-09-15	Processing	2024-09-16	150	2
1003	2024-09-20	Delivered	2024-09-21	300	3

(Payment_ref)

Order_ID (PK)	Payment_ref (Unique)
1000	PAY01234
1001	PAY54321
1002	PAY98765
1003	PAY24680

(tracking_no)

Order_ID (PK)	Tracking_nr (Unique)
1000	TRK00314
1001	TRK12345
1002	TRK67890
1003	TRK13579

(Review) Primary key (User_ID, Product_ID)

User_ID (FK)	Product_ID (FK)	Stars	Text
1	3001	5	Excellent product!
2	3002	3	It was okay.
3	3003	4	Good value for money

(Product)

Product_ID (PK)	Product_Title	Description	Price_excl_VAT	Stock_Qty	VAT_category (FK)	Sale_percentage	Is_Featured	Department_ID (FK)
3001	Smartphone	Latest model smartphone	500	10	1	10	True	4001
3002	Laptop	High-performance laptop	1200	5	1	15	False	4002
3003	NC_Headphones	Noise-cancelling headphones	150	20	1	5	False	4003
3004	WL_Headphones	Wireless headphones	150	20	1	5	True	4003

(VAT_Category)

ID (PK)	VAT percentage
1	25%
2	12%
3	6%
4	0%

(Ordered_Product)

Ordered_product_ID	Cost	Quantity	Name	Order_ID (FK)

(PK)				
1	250	1	Smartphone	1001
2	150	2	Laptop	1002
3	999	5	Rotary phone	1000
4	300	3	Headphones	1003

(Referenced_Product)

Ordered_Product_ID (PK) (FK)	Product_ID (FK)
1	3001
2	3002
4	3003

(Department)

Department_ID (PK)	Name	Description	Parent_ID (FK)
4000	Homepage	Welcome text	4000
4001	Electronics	…	4000
4002	Computers	PCs and laptops	4001
4003	Accessories	Accessories for computers	4002

(Keyword)

ID (PK)	Name
1	Mobile
2	Computer
3	Audio
4	Wireless

(Keyword_Assignments)

Keyword_ID (FK)	Product_ID (FK)
1	3001
2	3002
3	3003
3	3004
4	3004

SQL Task 3:

This is the SQL code for task 3, creating the relations:

```
CREATE TABLE `Department` (
  `Department_ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(100) NOT NULL,
  `Parent_ID` int NOT NULL,
  `Description` text,
  PRIMARY KEY (`Department_ID`),
  KEY `fk_parent_department` (`Parent_ID`),
  CONSTRAINT `fk_parent_department` FOREIGN KEY (`Parent_ID`) REFERENCES
`Department` (`Department_ID`)
);

CREATE TABLE `Keyword` (
  `Key_ID` int NOT NULL AUTO_INCREMENT,
  `Key` varchar(100) not NULL,
  PRIMARY KEY (`Key_ID`)
);

CREATE TABLE `VAT_Category` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `VAT_percentage` decimal(5,2) NOT NULL,
  PRIMARY KEY (`ID`)
);

CREATE TABLE `User` (
  `User_ID` int NOT NULL AUTO_INCREMENT,
```

```
  `Newsletter_permission` BOOLEAN NOT NULL DEFAULT FALSE,
  `Name` varchar(100) NOT NULL,
  `Street_Name` varchar(100) NOT NULL,
  `Password` varchar(255) NOT NULL,
  `Street_Number` varchar(10) NOT NULL,
  `City` varchar(50) NOT NULL,
  `Postal_Code` varchar(20) NOT NULL,
  PRIMARY KEY (`User_ID`)
);

CREATE TABLE `User_Emails` (
  `User_ID` int NOT NULL,
  `email` varchar(255) NOT NULL,
  PRIMARY KEY (`User_ID`),
  UNIQUE KEY `email` (`email`),
  CONSTRAINT `fk_user_email` FOREIGN KEY (`User_ID`) REFERENCES `User`
(`User_ID`)
);

CREATE TABLE `User_Phone_Numbers` (
  `User_ID` int NOT NULL,
  `Phone_Number` varchar(16) NOT NULL,
  PRIMARY KEY (`User_ID`),
  UNIQUE KEY `Phone_Number` (`Phone_Number`),
  CONSTRAINT `fk_user_phone` FOREIGN KEY (`User_ID`) REFERENCES `User`
(`User_ID`)
);

CREATE TABLE `User_PID` (
  `User_ID` int NOT NULL,
  `PID` varchar(16) NOT NULL,
  PRIMARY KEY (`User_ID`),
  UNIQUE KEY `PID` (`PID`),
  CONSTRAINT `fk_user_pid` FOREIGN KEY (`User_ID`) REFERENCES `User` (`User_ID`)
);

CREATE TABLE `Order` (
  `Order_ID` int NOT NULL AUTO_INCREMENT,
  `Date` date NOT NULL,
  `Status` varchar(50) NOT NULL,
  `Last_Change` date NOT NULL,
  `Cost` decimal(50,2) NOT NULL,
  `User_ID` int NOT NULL,
  PRIMARY KEY (`Order_ID`),
  KEY `fk_order_user` (`User_ID`),
  CONSTRAINT `fk_order_user` FOREIGN KEY (`User_ID`) REFERENCES `User`
(`User_ID`)
);

CREATE TABLE `Product` (
  `Product_ID` int NOT NULL AUTO_INCREMENT,
```

```
  `Product_Title` varchar(255) NOT NULL,
  `Description` text,
  `Price_without_VAT` decimal(10,2) NOT NULL,
  `Stock_qty` int NOT NULL,
  `Sale_percentage` decimal(5,2) NOT NULL,
  `VAT_Category` int NOT NULL,
  `Department_ID` int NOT NULL,
  `Is_featured` BOOLEAN NOT NULL,
  PRIMARY KEY (`Product_ID`),
  KEY `fk_product_vatcategory` (`VAT_Category`),
  KEY `fk_Department_ID` (`Department_ID`),
  CONSTRAINT `fk_Department_ID` FOREIGN KEY (`Department_ID`) REFERENCES
`Department` (`Department_ID`),
  CONSTRAINT `fk_product_vatcategory` FOREIGN KEY (`VAT_Category`) REFERENCES
`VAT_Category` (`ID`)
);

CREATE TABLE `Review` (
  `Stars` int NOT NULL,
  `Text` text,
  `Product_ID` int NOT NULL,
  `User_ID` int NOT NULL,
  PRIMARY KEY (`Product_ID`,`User_ID`),
  KEY `fk_review_user` (`User_ID`),
  CONSTRAINT `fk_review_product` FOREIGN KEY (`Product_ID`) REFERENCES `Product`
(`Product_ID`),
  CONSTRAINT `fk_review_user` FOREIGN KEY (`User_ID`) REFERENCES `User`
(`User_ID`),
  CONSTRAINT `Review_chk_1` CHECK ((`Stars` between 1 and 5))
);

CREATE TABLE `tracking_no` (
  `order_id` int NOT NULL,
  `tracking_no` varchar(100) NOT NULL,
  PRIMARY KEY (`order_id`),
  UNIQUE KEY `tracking_no` (`tracking_no`),
  CONSTRAINT `fk_tracking_no` FOREIGN KEY (`order_id`) REFERENCES `Order`
(`Order_ID`)
);

CREATE TABLE `Payment_ref` (
  `Order_ID` int NOT NULL,
  `Payment_ref` varchar(100) NOT NULL,
  PRIMARY KEY (`Order_ID`),
  UNIQUE KEY `Payment_ref` (`Payment_ref`),
  CONSTRAINT `fk_paymentref_order` FOREIGN KEY (`Order_ID`) REFERENCES `Order`
(`Order_ID`)
);

CREATE TABLE `Keyword_Assignments` (
  `Keyword_ID` int NOT NULL,
```

```
  `Product_ID` int NOT NULL,
  PRIMARY KEY (`Keyword_ID`, `Product_ID`),
  CONSTRAINT `Keyword_Assignments_ibfk_1` FOREIGN KEY (`Keyword_ID`) REFERENCES
`Keyword` (`Key_ID`),
  CONSTRAINT `Keyword_Assignments_ibfk_2` FOREIGN KEY (`Product_ID`) REFERENCES
`Product` (`Product_ID`)
);

CREATE TABLE `Ordered_Product` (
  `Ordered_Product_ID` int NOT NULL AUTO_INCREMENT,
  `Quantity` int NOT NULL,
  `Order_ID` int NOT NULL,
  `Cost` decimal(50,2) NOT NULL,
  `Name` varchar(255) NOT NULL,
  PRIMARY KEY (`Ordered_Product_ID`),
  KEY `fk_orderedproduct_order` (`Order_ID`),
  CONSTRAINT `fk_orderedproduct_order` FOREIGN KEY (`Order_ID`) REFERENCES
`Order` (`Order_ID`)
);

CREATE TABLE `Referenced_Product` (
  `Ordered_Product_ID` int NOT NULL,
  `Product_ID` int NOT NULL,
  PRIMARY KEY (`Ordered_Product_ID`),
  CONSTRAINT `fk_referenced_ordered_product` FOREIGN KEY (`Ordered_Product_ID`)
REFERENCES `Ordered_Product` (`Ordered_Product_ID`),
  CONSTRAINT `fk_referenced_product` FOREIGN KEY (`Product_ID`) REFERENCES
`Product` (`Product_ID`)
);
```

Tables are created in a sequence that respects foreign key dependencies.
No table tries to reference another table that does not yet exist.

Task 4.

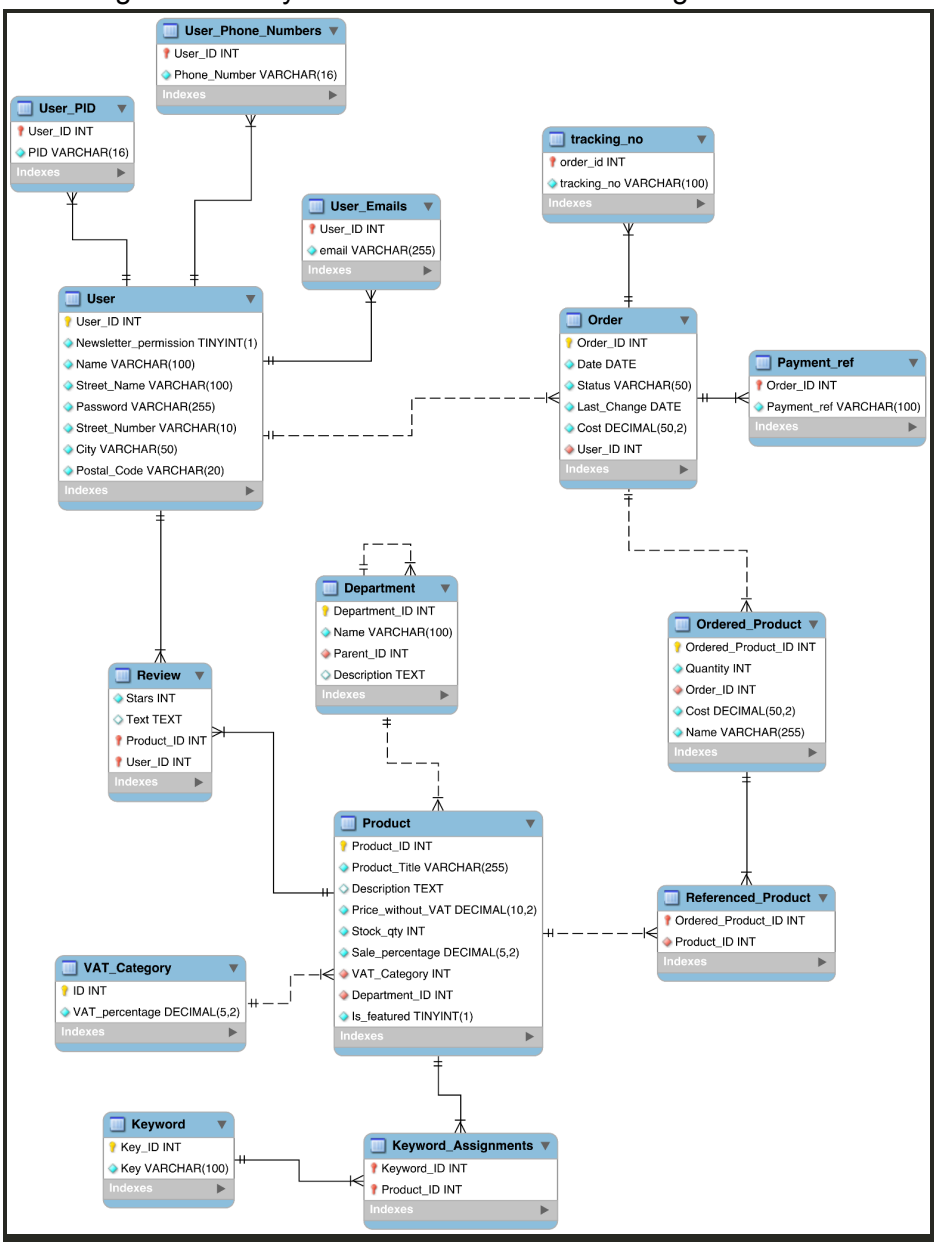
SQL-kod för att skapa 8 departments, varav 3 är “top departments”, 10 produkter, 2 användare, 2 reviews och en order.

```
ALTER TABLE Department MODIFY Parent_ID INT;

INSERT INTO Department (`Department_ID`, `Name`, `Description`,
`Parent_ID`)
VALUE
(9, 'Homepage', 'Welcome to AltOnline, where you can buy anything',
NULL);
```

```sql
UPDATE Department
SET Parent_ID = 9
WHERE Department_ID = 9;

ALTER TABLE Department MODIFY Parent_ID INT NOT NULL;

INSERT INTO Department (`Department_ID`, `Name`, `Description`,
`Parent_ID`)
VALUES
(1, 'Electronics', 'A selection of fine electronics', 9),
(2, 'Building material', 'A selection of fine building materials', 9),
(3, 'Wood', 'blabla', 2),
(4, 'Nails', 'blabla', 2),
(5, 'Machines', 'blabla', 2),
(6, 'Phones', 'blabla', 1),
(7, 'Chargers', 'blabla', 1),
(8, 'Computers', 'blabla', 1);

INSERT INTO VAT_Category (`ID`, `VAT_percentage`)
VALUES
(1, 25),
(2, 12),
(3, 6),
(4, 0);
Insert INTO Product (`Product_ID`, `Product_Title`, `Description`,
`Price_without_VAT`, `Stock_qty`, `VAT_Category`, `Sale_percentage`,
`Department_ID`, `Is_featured`)
VALUES
(1, 'Iphone X',          Null, 5500, 25,  1, 0, 6, 1),
(2, 'Samsung Galaxy S9', Null, 1200, 13,  1, 0, 6, 0),
(3, 'Nail 2mm',          Null, 0.35, 250, 1, 0, 4, 0),
(4, 'Nail 5mm',          Null, 0.45, 300, 1, 0, 4, 0),
(5, 'Nail 10mm',         Null, 0.75, 125, 1, 0, 4, 0),
(6, 'Screw driver',      Null, 1400, 150, 1, 0, 2, 1),
(7, 'Hammer',            Null, 450,  30,  1, 0, 2, 0),
(8, 'Wood 20x200',       Null, 200,  100, 1, 5.0,  3, 0),
(9, 'Wood 20x400',       Null, 200,  99,  1, 0, 3, 0),
(10, 'Wood 40x200',      Null, 130,  67,  1, 10.0, 3, 1);

INSERT INTO User (`User_ID`, `Newsletter_permission`, `Name`,
`Password`, `Street_Name`, `Street_Number`, `City`, `Postal_Code`)
VALUES
(1, 0, 'Lina', 'tacos123', 'Dekangatan', '56', 'Uppsala', '75645'),
(2, 1, 'Hanna', 'Sessan02 ', 'Prefektgatan', '48', 'Uppsala', '75642');
```

```sql
INSERT INTO Review (`Product_ID`, `User_ID`, `Stars`, `Text`)
VALUES
(1, 1,  3.4, 'Denna var ej bra'),
(2, 1, 4.9, 'Supernöjd' );

INSERT INTO `Order` (`Order_ID`, `Date`, `Status`, `Last_Change`,
`Cost`, `User_ID`) VALUES  (1, '2024-10-02', 'shipped', '2024-10-02',
5500, 1);

INSERT INTO Ordered_product (`Ordered_Product_ID`, `Cost`, `Quantity`,
`Name`, `Order_ID`)
VALUES
(1, 5500, 1, 'IPhone X', 1);
```

EER Diagram from MySQL Workbench Reverse Engineer.



Task 5.

Welcome text from the homepage:

```
SELECT Name, Description FROM Department WHERE Parent_ID = 9;
```

List of the top level departments with fields needed for the homepage:

```
SELECT Name, Description FROM Department WHERE Parent_ID = 9;
```

List of the featured products with fields needed for the homepage:

```
SELECT Product_ID, Product_Title, Description, Price_without_VAT,
(Price_without_VAT * (1 - Sale_percentage / 100)) AS
Current_Retail_Price, Sale_percentage FROM Product WHERE Is_Featured
= TRUE;
```

Given a product (the product where Product_ID=1 in our case), list all keyword-related products:

```
SELECT DISTINCT P.Product_ID, P.Product_Title, P.Description,
(P.Price_without_VAT * (1 - P.Sale_percentage / 100)) AS
Current_Retail_Price
FROM Product P
INNER JOIN Keyword_Assignments KA1 ON P.Product_ID =
KA1.Product_ID
INNER JOIN Keyword_Assignments KA2 ON KA1.Keyword_ID =
KA2.Keyword_ID
WHERE KA2.Product_ID = 1 AND P.Product_ID != 1;
```

• Given a department, list of all its products (title, short description, current retail price) with their average rating:

```
SELECT P.Product_Title, P.Description, (P.Price_without_VAT * (1 -
P.Sale_percentage / 100)) AS Current_Retail_Price, AVG(R.Stars) AS
Average_Rating
FROM Product P
LEFT JOIN Review R ON P.Product_ID = R.Product_ID
WHERE P.Department_ID = 6
GROUP BY P.Product_ID, P.Product_Title, P.Description,
P.Price_without_VAT, P.Sale_percentage;
```

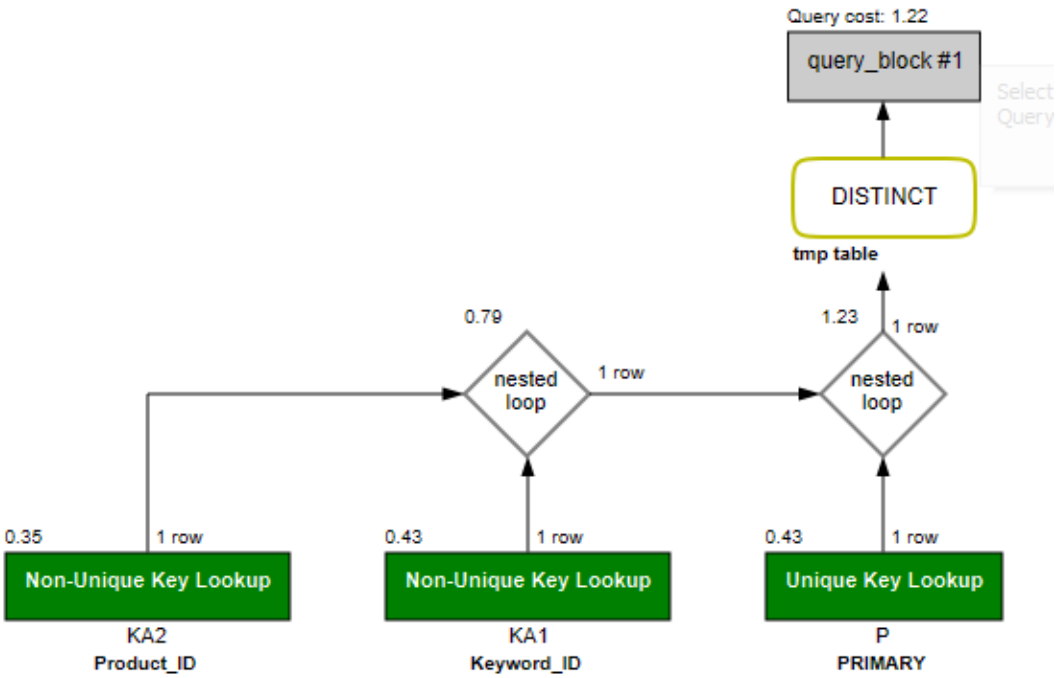
List of all products on sale sorted by the discount percentage (starting with the biggest discount):

```
SELECT Product_ID, Product_Title, Description, Sale_percentage,
(Price_without_VAT * (1 - Sale_percentage / 100)) AS
Current_Retail_Price FROM Product WHERE Sale_percentage > 0 ORDER
BY Sale_percentage DESC;
```

Task 6.

Starting of by looking at the more complex queries with many joins as they are the ones that benefit the most from indexing:

```
SELECT DISTINCT P.Product_ID, P.Product_Title, P.Description,
(P.Price_without_VAT * (1 - P.Sale_percentage / 100)) AS
Current_Retail_Price
FROM Product P
INNER JOIN Keyword_Assignments KA1 ON P.Product_ID =
KA1.Product_ID
INNER JOIN Keyword_Assignments KA2 ON KA1.Keyword_ID =
KA2.Keyword_ID
WHERE KA2.Product_ID = 1 AND P.Product_ID != 1;
```



This particular query is quite costly, which is a good starting point for the indexing.

Lets analyze it by running an EXPLAIN query.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	KA2	NULL	ref	Keyword_ID,Product_ID	Product_ID	4	const	1	100.00	Using temporary
1	SIMPLE	KA1	NULL	ref	Keyword_ID,Product_ID	Keyword_ID	4	ht24_1_project_group_35.KA2.Keyword_ID	1	100.00	Using where
1	SIMPLE	P	NULL	eq_ref	PRIMARY	PRIMARY	4	ht24_1_project_group_35.KA1.Product_ID	1	100.00	NULL

Here we are able to see where the indices should be placed, taking this into consideration we have to look at the indices that could improve this query.
We know that Hashing exponentially improves comparative operations, so in this case the biggest improvement would come from Binary Tree with major improvements to sorting, ordering and finding ranges. We would apply BTree indexes on the necessary columns given from the EXPLAIN, not on the whole table as that would increase cost and take more space while giving diminishing returns. Running this SQL:

```
CREATE INDEX idx_product_id_btree ON Product(Product_ID) USING BTREE;
```

Same for the other columns shown in the EXPLAIN, this is the updated result:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	KA2	NULL	ref	Keyword_ID,idx_keyword_product_btree,idx_k...	idx_keyword_product_btree	4	const	1	100.00	Using index; Using temporary
1	SIMPLE	KA1	NULL	ref	Keyword_ID,idx_keyword_product_btree,idx_k...	Keyword_ID	4	ht24_1_project_group_35.KA2.Keyword_ID	1	100.00	Using where
1	SIMPLE	P	NULL	eq_ref	PRIMARY,idx_product_id_btree	PRIMARY	4	ht24_1_project_group_35.KA1.Product_ID	1	100.00	NULL

By using ANALYZE TABLE we can also see if there were any errors by using the BTree index:

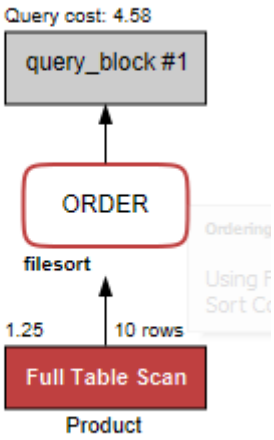
Table	Op	Msg_type	Msg_text
ht24_1_project_group_35.Product	analyze	status	OK

Table	Op	Msg_type	Msg_text
ht24_1_project_group_35.Keyword_Assignments	analyze	status	OK

The optimization was successful, although the tables are too small to see a big difference in execution time.

Next one that we are going to optimize, it has

```
SELECT Product_ID, Product_Title, Description, Sale_percentage,
(Price_without_VAT * (1 - Sale_percentage / 100)) AS
Current_Retail_Price
FROM Product
WHERE Sale_percentage > 0
ORDER BY Sale_percentage DESC;
```



Running explain gives:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	Product	NULL	ALL	NULL	NULL	NULL	NULL	10	33.33	Using where; Using filesort

So we add an index using the same logic as before:

```
CREATE INDEX idx_sale_percentage_btree ON Product (Sale_percentage) USING BTREE;
```

Worked as but the return was not as great as the last query.

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	Product	NULL	ALL	idx_sale_percentage_btree,idx2_sale_percenta...	NULL	NULL	NULL	10	100.00	Using where; Using filesort

Because the table was the same as earlier which was expected for this optimization.

Task 7.

In order to use the scripts the user must input credentials under “USERNAME” and “PASSWORD”, also pymysql and sshtunnel must be installed. Only thing that's added to the default connection code is a function that contains SQL code to fit the criteria of the task, with a simple user input in both scripts.

```python
import pymysql
from sshtunnel import SSHTunnelForwarder

tunnel = SSHTunnelForwarder(
    ('fries.it.uu.se', 22),
    ssh_username='USERNAME',
    ssh_password='PASSWORD',
    remote_bind_address=('127.0.0.1', 3306)
)
tunnel.start()

connection = pymysql.connect(
    host='127.0.0.1',
    user='ht24_1_group_35',
    password='pasSWd_35',
    port=tunnel.local_bind_port,
    database='ht24_1_project_group_35'
)

def list_items(department_id):
    cur = connection.cursor()
    cur.execute("SELECT COUNT(*) FROM Department WHERE Parent_ID = %s", (department_id,))
    if cur.fetchone()[0] == 0:
        cur.execute("SELECT Product_ID, Product_Title, (Price_without_VAT * (1 - Sale_percentage / 100)) FROM Product WHERE Department_ID = %s", (department_id,))
        for row in cur.fetchall():
            print(row)
    else:
        cur.execute("SELECT Department_ID, Description FROM Department WHERE Parent_ID = %s", (department_id,))
        for row in cur.fetchall():
            print(row)
    cur.close()

list_items(int(input("Enter the department ID: ")))

connection.close()
tunnel.stop()
```

```python
import pymysql
from sshtunnel import SSHTunnelForwarder

tunnel = SSHTunnelForwarder(
    ('fries.it.uu.se', 22),
    ssh_username='USERNAME',
    ssh_password='PASSWORD',
    remote_bind_address=('127.0.0.1', 3306)
)
tunnel.start()

connection = pymysql.connect(
    host='127.0.0.1',
    user='ht24_1_group_35',
    password='pasSWd_35',
    port=tunnel.local_bind_port,
    database='ht24_1_project_group_35'
)

def update_discount(product_id):
    cur = connection.cursor()
    cur.execute("SELECT Sale_percentage FROM Product WHERE Product_ID = %s", (product_id,))
    current_discount = cur.fetchone()
    if current_discount:
        print(f"Current Discount: {current_discount[0]}")
        new_discount = float(input("Enter new discount: "))
        cur.execute("UPDATE Product SET Sale_percentage = %s WHERE Product_ID = %s", (new_discount, product_id))
        connection.commit()
        print("Discount updated")
    else:
        print("Product not found")
    cur.close()

update_discount(int(input("Enter the product ID: ")))

connection.close()
tunnel.stop()
```