AltOnline

Database design 1
1DL301 - Group 35

Pavan Divi	pkd122470@gmail.com
Rasmus Hammar	rasmus.hammar.1670@student.uu.se
Akshath Nair	akshathnair8@gmail.com
Yeswasri Sivarathri	yeswasri.sivarathri.4977@student.uu.se
Somu Susmitha	somususmitha45@gmail.com

ER diagram---- 3
Normalization---- 4
1NF----4
2NF----5
3NF----5
SQL code---- 6
Tables----6
Populate database---- 8
Department----8
Products---- 9
Keywords----11
Customers---- 11
Orders---- 12
Products ordered----14
Reviews---- 15
Database schema---- 20
Queries---- 20
Homepage welcome text----20
List top level department----20
Featured products---- 20
Product keywords----21
Department products----21
Products on sale---- 21
Top ten best selling products----22
Indexes---- 22
EXPLAIN before----22
Index creation----23
EXPLAIN after---- 23
BTree and Hash indices----24
Python programs---- 26
mimer_department_products.py---- 26
mimer_change_discount.py---- 29
sshtunnel.py----32

ER diagram

In Milestone 1, we designed the ER diagram for the system by identifying all entities, their attributes, primary keys, derived attributes, multivalued attributes, and relationships. The main entities are DEPARTMENT, PRODUCT, CUSTOMER and ORDER. Unique attributes were underlined, multivalued attributes were shown with double ovals, and derived attributes with dashed ovals. The ER diagram (Figure 1) represents the complete conceptual structure of the system and shows how all entities are related before moving to normalization.

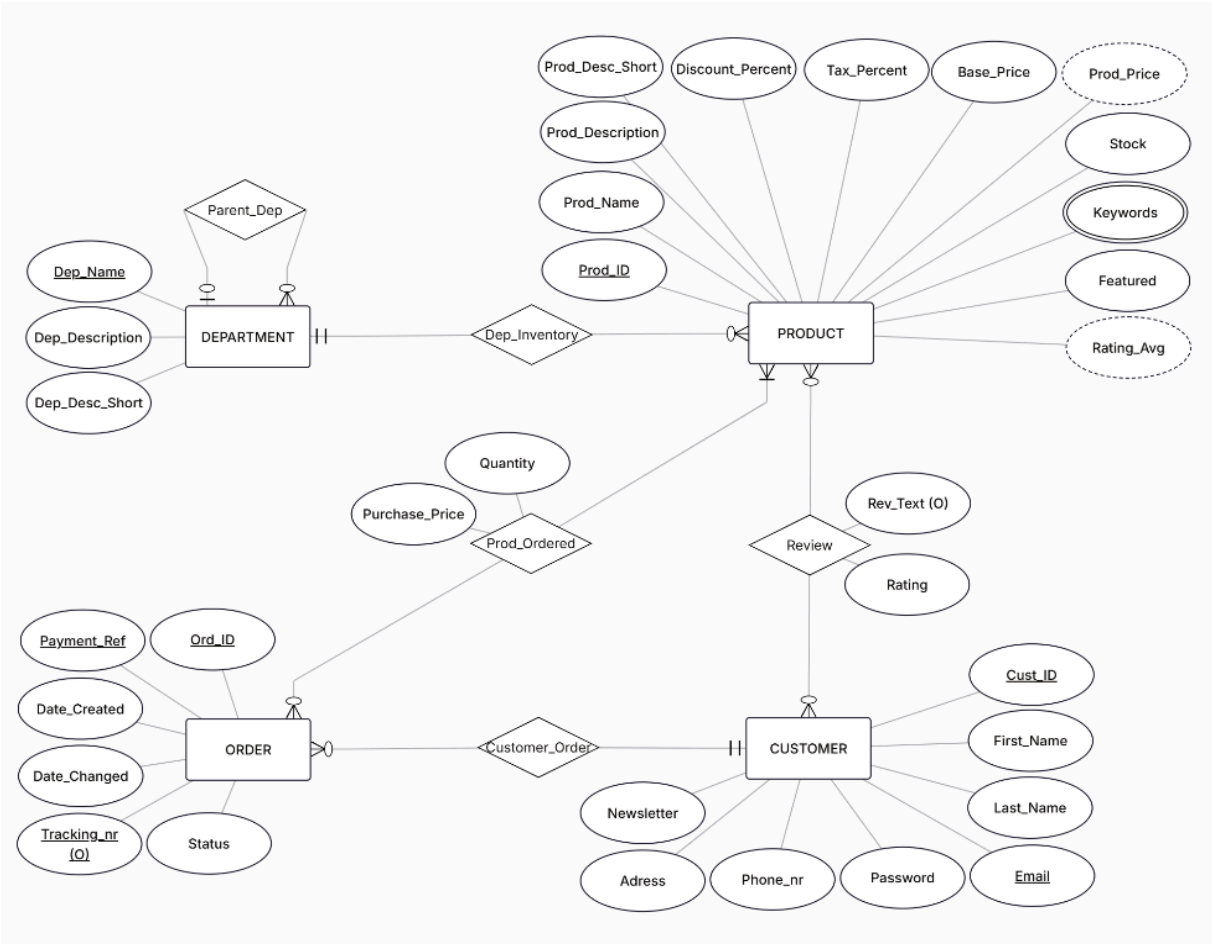

Figure 1: Entity relation diagram for the AltOnline database.

Normalization

In Milestone 2, we converted the ER diagram into normalized tables following 1NF, 2NF, and 3NF steps.

- 1NF: Removed multivalued and derived attributes (e.g., Keywords, Rating_Avg), and ensured all values were atomic. (Figure 2)
- 2NF: Checked composite keys (for e.g in Order_Items) and made sure every attribute depends on the whole key. (Figure 3)
- 3NF: Removed transitive dependencies (as there were no transitive dependencies we didn't made any changes so the 3NF table remains same as 2NF) and ensured all attributes depend only on the primary key (e.g., replacing Dep_Name as PK with Dep_ID).(Figure 3)

1NF

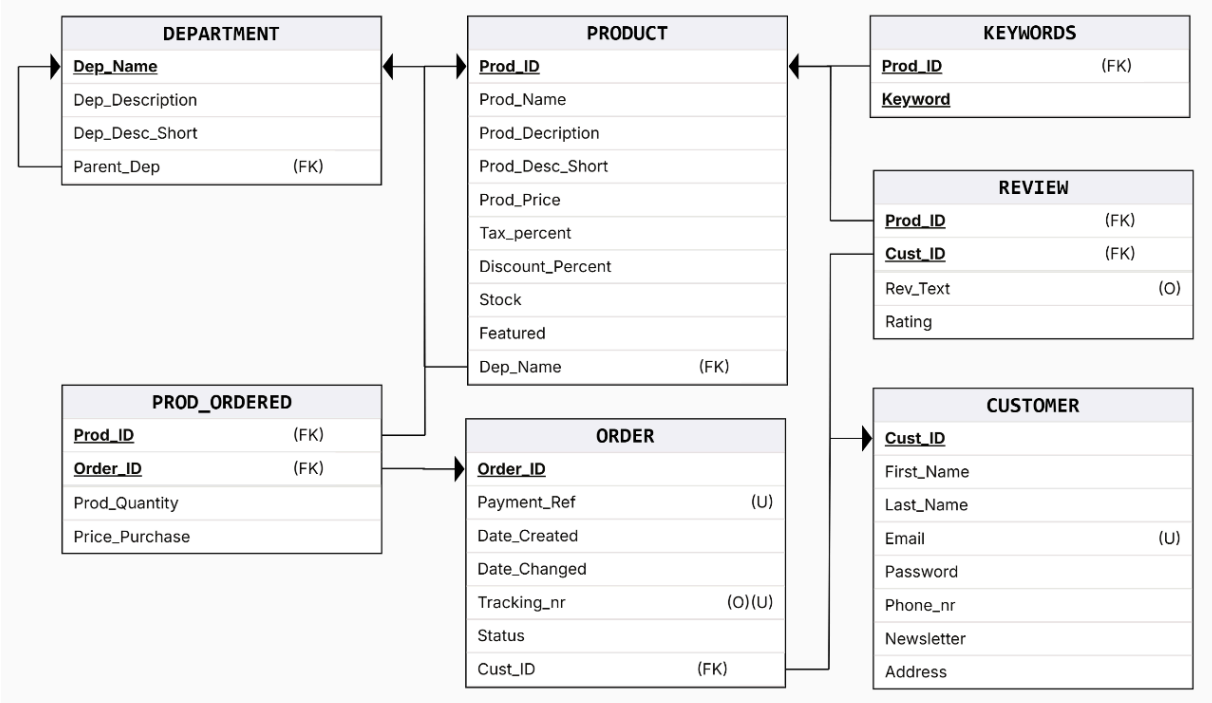

Figure 2: First normal form.

2NF

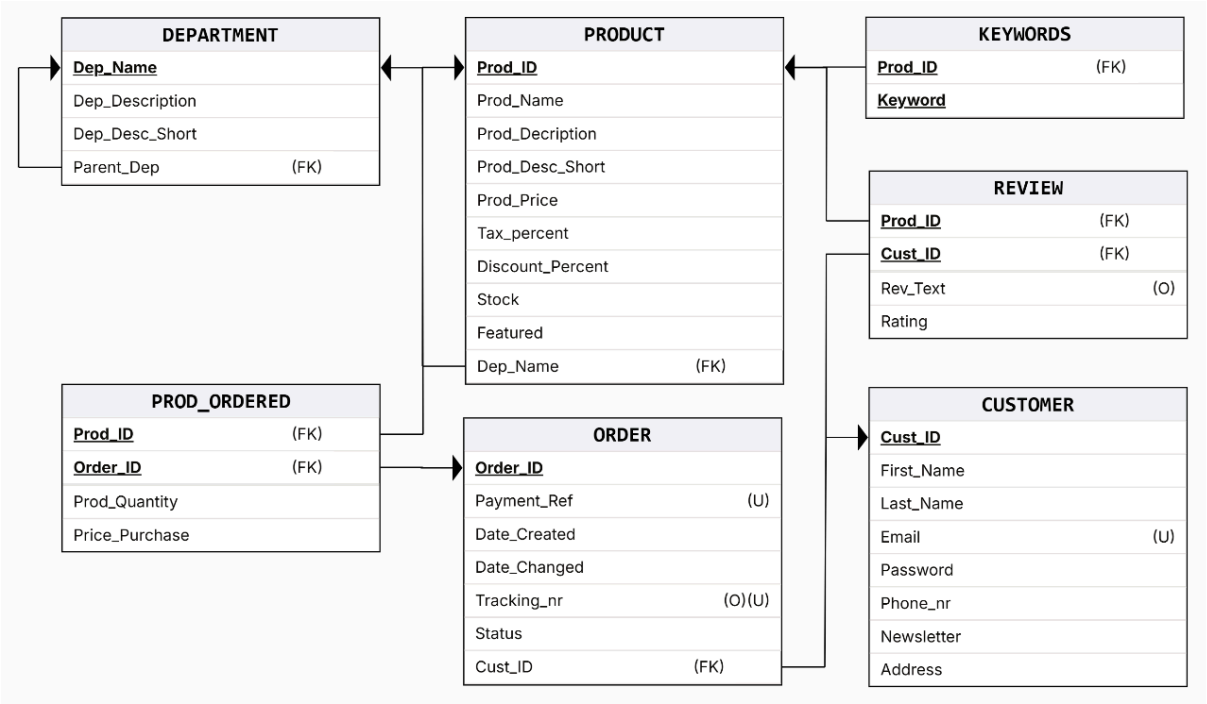

Figure 3: Second normal form.

3NF

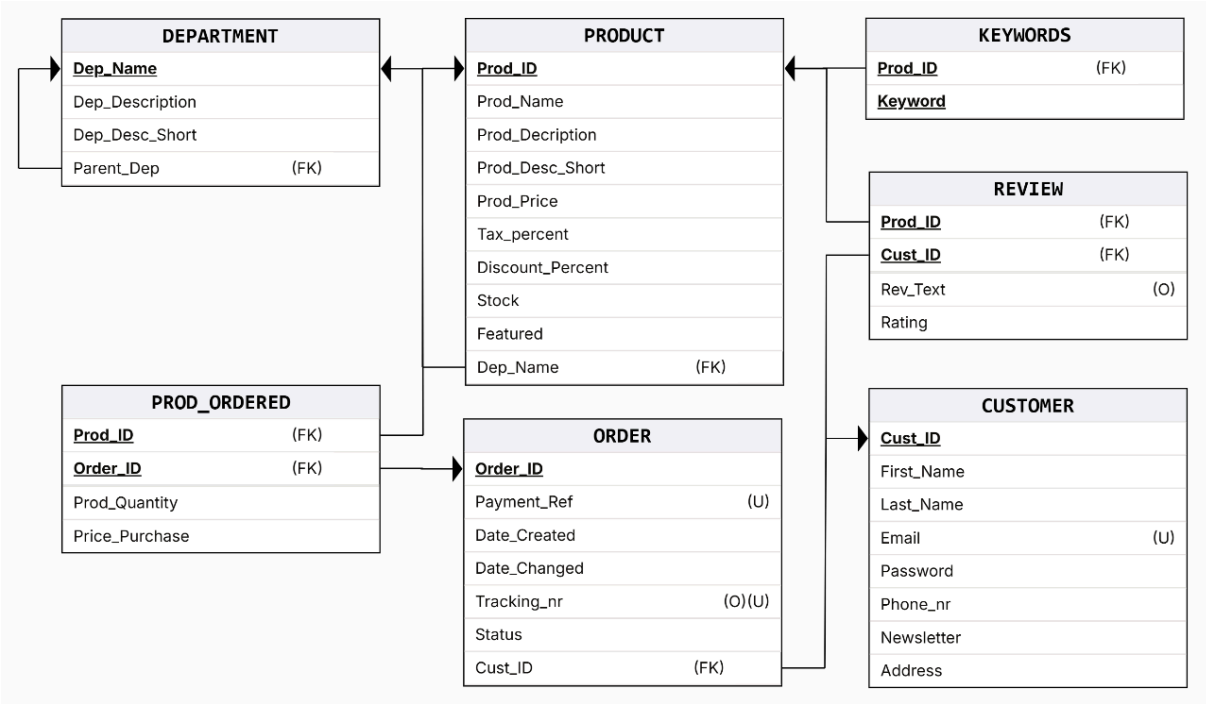

Figure 4: Third normal form.

SQL code

Tables

```
CREATE TABLE DEPARTMENT
(
    Dep_Name VARCHAR(25) NOT NULL,
    Dep_Description VARCHAR(10000),
    Dep_Desc_Short VARCHAR(500),
    Parent_Dep VARCHAR(25),
    PRIMARY KEY(Dep_Name),
    FOREIGN KEY(Parent_Dep) REFERENCES DEPARTMENT(Dep_Name)
);

CREATE TABLE PRODUCT
(
    Prod_ID INT NOT NULL,
    Prod_Name VARCHAR(100) NOT NULL,
    Prod_Description VARCHAR(10000),
    Prod_Desc_Short VARCHAR(200),
    Prod_Price DECIMAL(10, 2) NOT NULL,
    Tax_Percent INT NOT NULL,
    Discount_Percent INT NOT NULL,
    Stock INT NOT NULL,
    Featured BOOLEAN DEFAULT FALSE,
    Dep_Name VARCHAR(25),
    PRIMARY KEY(Prod_ID),
    FOREIGN KEY(Dep_Name) REFERENCES DEPARTMENT(Dep_Name)
);

CREATE TABLE KEYWORDS
(
    Prod_ID INT NOT NULL,
    Keyword VARCHAR(100) NOT NULL,
    PRIMARY KEY (Prod_ID, Keyword),
    FOREIGN KEY (Prod_ID) REFERENCES PRODUCT(Prod_ID) ON DELETE CASCADE
);

CREATE TABLE CUSTOMER
(
    Cust_ID CHAR(12) NOT NULL, -- Personnummer
    First_Name VARCHAR(100),
    Last_Name VARCHAR(100),
```

```
    Email VARCHAR(255) NOT NULL UNIQUE,
    Password VARCHAR(255) NOT NULL,
    Phone_nr VARCHAR(50),
    Newsletter BOOLEAN DEFAULT FALSE,
    Address VARCHAR(200),
    PRIMARY KEY(Cust_ID)
);

CREATE TABLE REVIEW (
    Prod_ID INT NOT NULL,
    Cust_ID CHAR(12) NOT NULL,
    Rating INT NOT NULL CHECK (Rating BETWEEN 1 AND 5),
    Rev_Text VARCHAR(500),
    Rev_Date DATE,
    PRIMARY KEY(Prod_ID, Cust_ID),
    FOREIGN KEY(Prod_ID) REFERENCES PRODUCT(Prod_ID) ON DELETE CASCADE,
    FOREIGN KEY(Cust_ID) REFERENCES CUSTOMER(Cust_ID) ON DELETE CASCADE
);

CREATE TABLE ORDERS
(
    Order_ID INT NOT NULL,
    Cust_ID CHAR(12) NOT NULL,
    Date_Created DATE,
    Date_Changed DATE,
    Status VARCHAR(30),  -- ENUM('new','open','dispatched') DEFAULT
'new',
    Payment_Ref CHAR(64),
    Tracking_nr VARCHAR(255),
    PRIMARY KEY(Order_ID),
    FOREIGN KEY(Cust_ID) REFERENCES CUSTOMER(Cust_ID)
);

CREATE TABLE PROD_ORDERED
(
    Prod_ID INT NOT NULL,
    Order_ID INT NOT NULL,
    Prod_Quantity INT NOT NULL,
    Price_Purchase DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (Order_ID, Prod_ID),
    FOREIGN KEY(Prod_ID) REFERENCES PRODUCT(Prod_ID),
    FOREIGN KEY(Order_ID) REFERENCES ORDERS(Order_ID)
);
```

Populate database

Department

```
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Home', 'All departments', 'Homepage', NULL);
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Electronics', 'All electronic devices and gadgets',
'Electronics', Home);
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Mobile Phones', 'Smartphones and mobile accessories', 'Mobiles',
'Electronics');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Laptops', 'All types of laptops and accessories', 'Laptops',
'Electronics');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Gaming computers', 'All types of high-end PC hardware', 'Gaming
stuff', 'Electronics');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Home Appliances', 'Appliances for home use', 'Home Appliances',
Home);
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Heating', 'Appliances for heating your home', 'Radiators and
stuff', 'Home Appliances');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
```

```
    ('Cleaning', 'Appliances for keeping your home clean', 'Cleaning
supplies', 'Home Appliances');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Cooling', 'Appliances for staying cool in summer', 'AC and
stuff', 'Home Appliances');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Groceries', 'Food and all other food related stuff', 'Groceries',
Home);
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Crackers and Waffers', 'Munching Items', 'Snacks', 'Groceries');
INSERT INTO DEPARTMENT
    (Dep_Name, Dep_Description, Dep_Desc_Short, Parent_Dep)
VALUES
    ('Hot and Cold Drinks', 'Drinking Stuff', 'Bevarages',
'Groceries');
```

Products

```
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (1, 'iPhone 15', 'Apple iPhone 15, 128GB, Silver', 'iPhone 15
128GB', 999.99, 18, 5, 50, TRUE, 'Mobile Phones');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (2, 'Samsung Galaxy S23', 'Samsung flagship phone, 256GB', 'Galaxy
S23', 899.99, 18, 10, 30, TRUE, 'Mobile Phones');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (3, 'Dell XPS 13', 'Dell XPS 13 Laptop, Intel i7, 16GB RAM', 'XPS
13 Laptop', 1199.99, 18, 8, 20, FALSE, 'Laptops');
INSERT INTO PRODUCT
```

```
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (4, 'Dyson V15 Vacuum', 'Dyson cordless vacuum cleaner', 'Dyson
Vacuum', 599.99, 18, 15, 15, FALSE, 'Cleaning');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (7, 'Dyson Ultra AC', 'Dyson high capacity AC for large spaces',
'Dyson AC', 599.99, 18, 15, 15, FALSE, 'Cooling');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (11, 'ASUS Zenbook 14', 'ASUS Zenbook 14 Laptop, Intel core ultra
7, 32GB RAM', 'ASUS Zenbook 14 Laptop', 11990.99, 18, 15, 20, TRUE,
'Laptops');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (6, 'Sony WH-1000XM5', 'Sony WH-1000XM5 Wireless Noise-Cancelling
Headphones, 30h battery life', 'Sony WH-1000XM5 Headphones', 399.99,
20, 12, 100, FALSE, 'Gaming computers');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (8, 'LG UltraGear 34"', 'LG UltraGear 34-inch Curved Gaming
Monitor, 165Hz, QHD', 'LG UltraGear 34"', 899.00, 22, 10, 40, FALSE,
'Gaming computers');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (10, 'Oreo Biscuits', 'Delicious chocolate sandwich cookies',
'Oreo', 3.99, 5, 10, 100, false, 'Groceries');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
```

```
    (12, 'Lays Potato Chips', 'Crispy salted potato chips', 'Lays
Chips', 2.49, 5, 5, 150, false, 'Crackers and Waffers');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (15, 'Coca Cola 500ml', 'Refreshing carbonated soft drink', 'Coca
Cola', 1.49, 5, 0, 200, true, 'Hot and Cold Drinks');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (19, 'Pepsi 500ml', 'Popular cola beverage', 'Pepsi', 1.39, 5, 0,
180, true, 'Hot and Cold Drinks');
INSERT INTO PRODUCT
    (Prod_ID, Prod_Name, Prod_Description, Prod_Desc_Short, Prod_Price,
Tax_Percent, Discount_Percent, Stock, Featured, Dep_Name)
VALUES
    (20, 'Sprite 500ml', 'Lemon-lime soft drink', 'Sprite', 1.29, 5, 0,
160, false, 'Hot and Cold Drinks');
```

Keywords

```
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (1, 'Smartphone');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (1, 'Apple');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (2, 'Smartphone');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (2, 'Samsung');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (3, 'Laptop');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (3, 'Dell');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (4, 'Vacuum');
INSERT INTO KEYWORDS (Prod_ID, Keyword) VALUES (4, 'Home Appliance');
```

Customers

```
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
    ('100000000001', 'Alice', 'Smith', 'alice@example.com',
'password123', '5551234567', TRUE, '12 Elm Street');
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
```

```
    ('100000000002', 'Bob', 'Johnson', 'bob@example.com', 'pass456',
'5559876543', FALSE, '34 Oak Avenue');
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
    ('100000000011', 'Charlie', 'Williams', 'charlie82@example.com',
'charliePass', '5551112233', TRUE, '56 Maple Road');
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
    ('100000000012', 'Diana', 'Brown', 'diana@example.com',
'dianaSecure', '5552223344', FALSE, '78 Pine Lane');
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
    ('100000000005', 'Ethan', 'Davis', 'ethan@example.com', 'ethanKey',
'5553334455', TRUE, '90 Cedar Street');
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
    ('100000000006', 'Fiona', 'Miller', 'fiona@example.com',
'fionaLock', '5554445566', TRUE, '23 Birch Avenue');
INSERT INTO CUSTOMER
    (Cust_ID, First_Name, Last_Name, Email, Password, Phone_nr,
Newsletter, Address)
VALUES
    ('100000000007', 'George', 'Wilson', 'george@example.com',
'georgeSafe', '5555556677', FALSE, '45 Willow Drive');
```

Orders

```
INSERT INTO ORDERS
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10001, '100000000001', DATE'2025-12-01', DATE'2025-12-02', 'new',
'PAY00124548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK123');
INSERT INTO ORDERS
```

```
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10002, '100000000002', DATE'2025-12-02', DATE'2025-12-03', 'open',
'PAY00224548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK456');
INSERT INTO Orders
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10001, '100000000001', DATE '2025-12-01', DATE '2025-12-02',
'new', 'PAY00124548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK123');
INSERT INTO Orders
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10002, '100000000002', DATE '2025-12-02', DATE '2025-12-03',
'open', 'PAY00224548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK456');
INSERT INTO Orders
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10003, '100000000003', DATE '2025-12-03', DATE '2025-12-04',
'dispatched',
'PAY00324548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK789');
INSERT INTO Orders
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10004, '100000000004', DATE '2025-12-04', DATE '2025-12-05',
'new', 'PAY00424548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK101');
INSERT INTO Orders
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    (10005, '100000000001', DATE '2025-12-05', DATE '2025-12-06',
'open', 'PAY00524548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK102');
INSERT INTO Orders
```

```
    (Order_ID, Cust_ID, Date_Created, Date_Changed, Status,
Payment_Ref, Tracking_nr)
VALUES
    10006, '100000000002', DATE '2025-12-06', DATE '2025-12-07',
'dispatched',
'PAY00624548515151545554545454XXXXXXXXXXXXXXXXXXXXXXXXXXXX',
'TRACK103');
```

Products ordered

```
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (1, 10001, 1, 999.99);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (3, 10002, 1, 1199.99);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (2, 10001, 2, 899.99);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (12, 10004, 4, 1.49);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (10, 10001, 1, 3.99);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (15, 10003, 6, 1.49);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (11, 10005, 1, 3.99);
INSERT INTO PROD_ORDERED
    (Prod_ID, Order_ID, Prod_Quantity, Price_Purchase)
VALUES
    (3, 10006, 6, 1.49);
```

Reviews

```
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (1, '100000000001', 5, 'Amazing iPhone, love it!',
DATE'2025-12-02');
INSERT INTO Review
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (2, '100000000002', 4, 'Great phone but battery could be better',
DATE'2025-12-03');
INSERT INTO Review
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (3, '100000000001', 5, 'Excellent laptop for work and travel',
DATE'2025-12-02');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (6, '100000000007', 5, 'Noise cancellation is unbeatable, perfect
for travel.', DATE'2025-12-08');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (6, '100000000002', 4, 'Great sound quality but a bit pricey.',
DATE'2025-12-09');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (1, '100000000002', 4, 'Great phone but battery drains quicker than
expected.', DATE'2025-12-03');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (1, '100000000011', 5, 'Super smooth performance, Face ID works
flawlessly.', DATE'2025-12-28');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (1, '100000000012', 3, 'Good phone but too pricey for the
features.', DATE'2025-12-29');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
```

```
VALUES
    (2, '100000000001', 5, 'Fantastic display and smooth performance!',
DATE'2025-12-04');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (2, '100000000005', 5, 'Battery life is excellent, lasts all day
easily.', DATE'2025-12-30');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (2, '100000000006', 4, 'Camera is sharp but phone feels heavy.',
DATE'2025-12-31');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (3, '100000000002', 4, 'Solid performance but fan noise is
noticeable.', DATE'2025-12-07');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (3, '100000000007', 5, 'Compact design, perfect for business
travel.', DATE'2026-01-01');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (3, '100000000011', 4, 'Fast performance but limited ports.',
DATE'2026-01-02');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (4, '100000000001', 5, 'Cleans carpets and hardwood floors
effortlessly!', DATE'2025-12-08');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (4, '100000000002', 3, 'Good suction but battery life is shorter
than expected.', DATE'2025-12-09');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (4, '100000000012', 5, 'Lightweight and powerful, makes cleaning
easy.', DATE'2026-01-03');
```

```
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (4, '100000000005', 4, 'Great suction but charging takes too
long.', DATE'2026-01-04');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (7, '100000000001', 5, 'Keeps the whole living room cool even in
summer heat.', DATE'2025-12-10');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (7, '100000000002', 4, 'Strong cooling but a bit noisy at night.',
DATE'2025-12-11');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (7, '100000000006', 5, 'Cools the room quickly, very effective.',
DATE'2026-01-05');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (7, '100000000007', 3, 'Works fine but consumes a lot of
electricity.', DATE'2026-01-06');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (11, '100000000001', 5, 'Super fast laptop, handles multitasking
with ease.', DATE'2025-12-12');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (11, '100000000002', 4, 'Excellent build quality but quite
expensive.', DATE'2025-12-13');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (11, '100000000011', 5, 'Handles heavy software easily, great
productivity laptop.', DATE'2026-01-07');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
```

```
    (11, '100000000012', 4, 'Excellent keyboard but screen brightness
could be better.', DATE'2026-01-08');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (6, '100000000001', 5, 'Noise cancellation is incredible, perfect
for flights.', DATE'2025-12-14');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (6, '100000000005', 5, 'Crystal clear audio, perfect for music
lovers.', DATE'2026-01-09');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (6, '100000000006', 3, 'Good headphones but uncomfortable after
long use.', DATE'2026-01-10');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (8, '100000000001', 5, 'Immersive curved screen, gaming feels
amazing!', DATE'2025-12-16');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (8, '100000000002', 4, 'Beautiful visuals but takes up a lot of
desk space.', DATE'2025-12-17');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (8, '100000000007', 5, 'Amazing refresh rate, perfect for
competitive gaming.', DATE'2026-01-11');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (8, '100000000011', 4, 'Great visuals but expensive.',
DATE'2026-01-12');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (10, '100000000001', 5, 'Classic taste, always delicious with
milk.', DATE'2025-12-18');
INSERT INTO REVIEW
```

```
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (10, '100000000002', 4, 'Great cookies but a bit too sweet.',
DATE'2025-12-19');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (10, '100000000012', 5, 'Classic cookies, my kids love them.',
DATE'2026-01-13');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
VALUES
    (10, '100000000005', 4, 'Tasty but packaging could be better.',
DATE'2026-01-14');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
    VALUES
    (12, '100000000001', 5, 'Crispy and perfectly salted, my favorite
snack.', DATE'2025-12-20');
INSERT INTO REVIEW
    (Prod_ID, Cust_ID, Rating, Rev_Text, Rev_Date)
    VALUES
    (12, '100000000002', 3, 'Good chips but too oily for my taste.',
DATE'2025-12-21');
```

Database schema

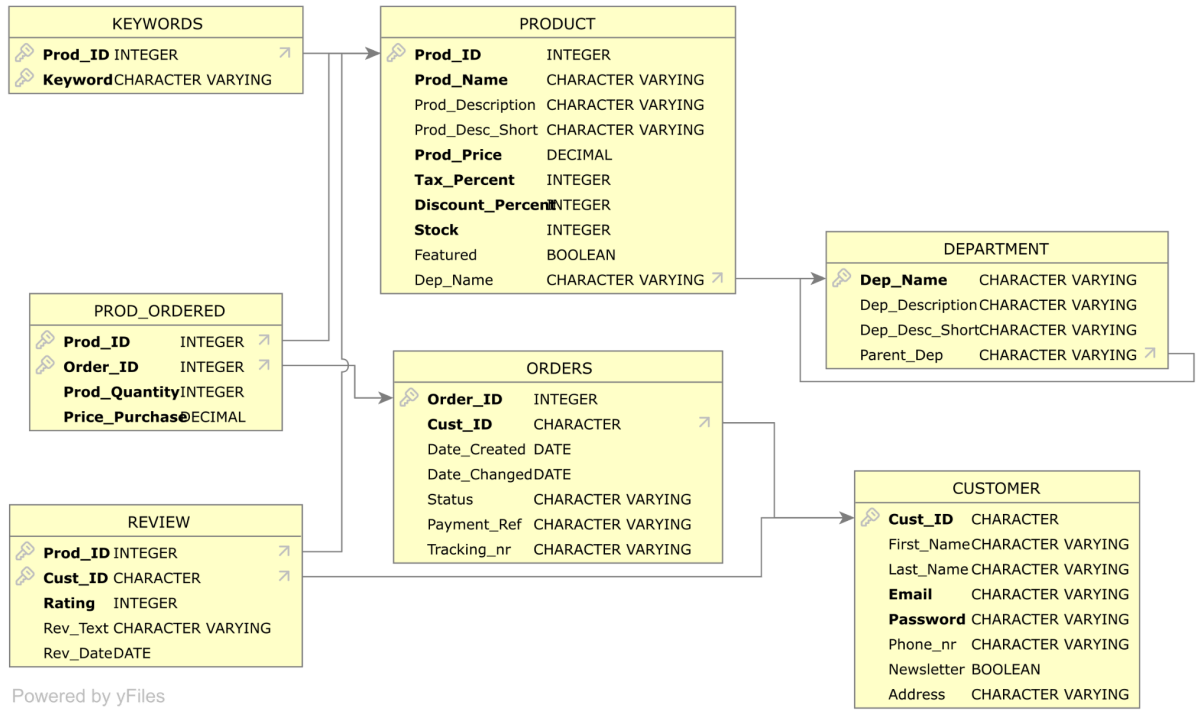

Figure 5: Database schema from MIMER.

Queries

Homepage welcome text

```
SELECT Dep_Description
FROM Department
WHERE Parent_Dep IS NULL;
```

List top level department

```
SELECT
    Dep_Name AS Title,
    Dep_Desc_Short AS Short_Description
FROM Department
WHERE Parent_Dep IS NULL
ORDER BY Dep_Name;
```

Featured products

```
SELECT
    Prod_ID,
    Prod_Name AS Title,
    Prod_Desc_Short AS Short_Description,
    ROUND(Prod_Price * (1 + Tax_Percent / 100) * (1 - Discount_Percent
/ 100), 2)
```

```
        AS Current_Retail_Price
FROM Product
WHERE Featured = true
ORDER BY Prod_Name;
```

Product keywords

```
SELECT
    p.Prod_ID,
    p.Prod_Name AS Title,
    p.Prod_Desc_Short AS Short_Description,
    ROUND(p.Prod_Price * (1 + p.Tax_Percent/100) * (1 -
p.Discount_Percent/100), 2)
        AS Current_Retail_Price,
    ROUND(AVG(r.Rating), 2) AS Avg_Rating
FROM Product p
LEFT JOIN Review r ON r.Prod_ID = p.Prod_ID
WHERE p.Dep_Name = ?
GROUP BY p.Prod_ID, p.Prod_Name, p.Prod_Desc_Short,
         p.Prod_Price, p.Tax_Percent, p.Discount_Percent
ORDER BY p.Prod_Name;
```

Department products

```
SELECT DISTINCT
    p2.Prod_ID,
    p2.Prod_Name,
    p2.Prod_Desc_Short,
    ROUND(p2.Prod_Price * (1 + p2.Tax_Percent/100) * (1 -
p2.Discount_Percent/100), 2)
        AS Retail_Price_With_Tax
FROM Keywords k1
JOIN Keywords k2 ON k1.Keyword = k2.Keyword
JOIN Product p2 ON p2.Prod_ID = k2.Prod_ID
WHERE k1.Prod_ID = ?
  AND k2.Prod_ID <> k1.Prod_ID
ORDER BY p2.Prod_Name;
```

Products on sale

```
SELECT
    Prod_ID,
    Prod_Name,
    Prod_Desc_Short,
    Discount_Percent,
```

```
    ROUND(Prod_Price * (1 + Tax_Percent/100) * (1 -
Discount_Percent/100), 2)
        AS Current_Retail_Price
FROM Product
WHERE Discount_Percent > 0
ORDER BY Discount_Percent DESC, Prod_Name;
```

Top ten best selling products

```
SELECT
    p.Prod_ID,
    p.Prod_Name,
    SUM(po.Prod_Quantity) AS Total_Sold
FROM Prod_Ordered po
JOIN Orders o ON po.Order_ID = o.Order_ID
JOIN Product p ON po.Prod_ID = p.Prod_ID
WHERE o.Date_Created >= CURRENT_DATE - INTERVAL '30' DAY
GROUP BY p.Prod_ID, p.Prod_Name
ORDER BY Total_Sold DESC
FETCH FIRST 10 ROWS ONLY;
```

Indexes

EXPLAIN before

Table 1: Homepage welcome text

ID	1	2
PARENT	0	1
OPERATION	select	index scan, table lookup
OPERATIONTYPE	(null)	leading keys
SCANORDER	(null)	1
ACC_COST	12	12
HITS	6	6
VISITS	12	12
TABLE	(null)	Department
ALIAS	(null)	(null)
INDEX	(null)	SQL_FOREIGN_KEY_0000049648
INDEXONLY	(null)	FALSE

Table 2: Featured products

ID	1	2	3
PARENT	0	1	2
OPERATION	select	sorting	full table scan
OPERATIONTYPE	(null)	temporary table order by	sequential
SCANORDER	(null)	(null)	1
ACC_COST	22	(null)	13
HITS	4	(null)	4
VISITS	17	4	13
TABLE	(null)	(null)	Product
ALIAS	(null)	(null)	(null)
INDEX	(null)	(null)	SQL_PRIMARY_KEY_0 000049654
INDEXONLY	(null)	(null)	(null)

Index creation

```
CREATE INDEX idx_product_name         ON product (prod_name);
CREATE INDEX idx_product_dep          ON product (dep_name);
CREATE INDEX idx_product_price        ON product (prod_price);
CREATE INDEX idx_product_featured     ON product (featured);

CREATE INDEX idx_product_dep_price    ON product (dep_name, prod_price);

CREATE INDEX idx_keywords_keyword     ON keywords (keyword);

CREATE INDEX idx_review_prod          ON review (prod_id);
CREATE INDEX idx_review_cust          ON review (cust_id);
CREATE INDEX idx_review_date          ON review (rev_date);

CREATE INDEX idx_customer_lastname    ON customer (last_name);
CREATE INDEX idx_customer_phone       ON customer (phone_nr);

CREATE INDEX idx_orders_cust          ON orders (cust_id);
CREATE INDEX idx_orders_status_date   ON orders (status, date_created);

CREATE INDEX idx_prod_ordered_prodid ON prod_ordered (prod_id);
```

EXPLAIN after

Table 3: Homepage welcome text with indices

ID	1	2
PARENT	0	1
OPERATION	select	index scan, table lookup
OPERATIONTYPE	(null)	leading keys
SCANORDER	(null)	1
ACC_COST	12	12
HITS	6	6
VISITS	12	12
TABLE	(null)	Department
ALIAS	(null)	(null)
INDEX	(null)	SQL_FOREIGN_KEY_0000049648
INDEXONLY	(null)	FALSE

Table 4: Featured products with indices.

ID	1	2	3
PARENT	0	1	2
OPERATION	select	sorting	index scan, table lookup
OPERATIONTYPE	(null)	temporary table order by	leading keys
SCANORDER	(null)	(null)	1
ACC_COST	18	(null)	9
HITS	4	(null)	4
VISITS	13	4	9
TABLE	(null)	(null)	Product
ALIAS	(null)	(null)	(null)
INDEX	(null)	(null)	idx_product_featured
INDEXONLY	(null)	(null)	FALSE

BTree and Hash indices

- A B-tree index stores values in a balanced tree structure, allowing the DBMS to perform fast quality searches, range searches, sorting, and prefix matching. It is the default general-purpose index.

- A hash index uses a hash table optimized for equality comparisons. It is faster for lookups but does not support range queries or ordering, because hashed values are not stored in sorted order.
- B-tree supports ‘=’, ’<’, ‘>’, ‘<=’, ‘>=’, ‘BETWEEN’, and ‘ORDER BY’. Where as Hash indexes only support ‘=’
- B-tree indexes are used in most cases while hash indexes are chosen only when dealing with frequent, high-performance equality lookups.

Python programs

mimer_department_products.py

```
import getpass

import mimerpy
from sshtunnel import SSHTunnel

group_name = "ht25_2_1dl301_group_35"
group_password = "pasSWd_35"


def program(mydb):
    mycursor = mydb.cursor()
    ## Get all department names and show user
    mycursor.execute("SELECT Dep_Name FROM DEPARTMENT ")
    for x in mycursor:
        print(x)

    ## Get user input department
    user_department = input("Select department to view products: ")

    ## Get list of leaf departments
    mycursor.execute("""
        SELECT Dep_Name
        FROM DEPARTMENT
        WHERE Parent_Dep IS NOT NULL
            AND Dep_Name NOT IN (
                SELECT Parent_Dep
                FROM DEPARTMENT
                WHERE Parent_Dep IS NOT NULL
            )
    """)
    leaf_departments = [row[0] for row in mycursor.fetchall()]

    ## Check if user department is leaf or not
    if user_department in leaf_departments:
        ### Leaf department -> list products
        print(f"Products in {user_department}:")
        mycursor.execute(
            """
            SELECT
```

```
                Prod_ID,
                Prod_Name,
                ROUND(Prod_Price * (1+Tax_Percent/100) *
(1-Discount_Percent/100), 2)
                AS Retail_Price
            FROM PRODUCT
            WHERE Dep_Name = '{}'
            """.format(user_department)
        )
        for x in mycursor:
            print(f"\t{x}")
    else:
        ### Not leaf -> list child departments
        print(f"Child departments in {user_department}:")
        mycursor.execute(
            """
            SELECT Dep_Name
            FROM DEPARTMENT
            WHERE Parent_Dep = '{}'
            """.format(user_department)
        )
        for x in mycursor:
            print(f"\t{x}")

    mycursor.close()


def db_connect():
    mydb = mimerpy.connect(dsn="DBIP22024", user=group_name,
password=group_password)

    program(mydb)

    mydb.close()


if __name__ == "__main__":
    ssh_username = input("Enter your Studium username: ")
    ssh_password = getpass.getpass("Enter your Studium password A: ")

    tunnel = SSHTunnel(ssh_username, ssh_password, "groucho.it.uu.se",
22)
    tunnel.start(
```

```
    local_host="127.0.0.1",
    local_port=13600,
    remote_host="127.0.0.1",
    remote_port=1360,
)

# Now the tunnel is ready, connect to DB
db_connect()

# Stop the tunnel
tunnel.stop()
```

mimer_change_discount.py

```
import getpass

import mimerpy
from sshtunnel import SSHTunnel

group_name = "ht25_2_1dl301_group_35"
group_password = "pasSWd_35"


def program(mydb):
    mycursor = mydb.cursor()

    ## Get Prod_ID from user
    user_prod_ID = input("Select Product (Prod_ID): ")

    ## Get product + discount
    mycursor.execute(
        """
        SELECT Prod_ID, Prod_Name, Discount_Percent
        FROM PRODUCT
        WHERE Prod_ID = {}
        """.format(user_prod_ID)
    )
    for x in mycursor:
        print(x)

    ## Ask user for new discount
    new_discount = input("New discount (%): ")

    ## Update discount
    if new_discount == "":
        ### Empty string -> no update (extend to check for int?)
        print("No changes made: ")
    else:
        ### int -> update discount -> commit transaction
        mycursor.execute(
            """
            UPDATE PRODUCT
            SET Discount_Percent = {}
            WHERE Prod_ID = {}
            """.format(new_discount, user_prod_ID)
        )
```

```
        mydb.commit()
        print("Update done: ")

    ## Confirm final value
    mycursor.execute(
        """
        SELECT Prod_ID, Prod_Name, Discount_Percent
        FROM PRODUCT
        WHERE Prod_ID = {}
        """.format(user_prod_ID)
    )
    for x in mycursor:
        print(x)

    mycursor.close()


def db_connect():
    mydb = mimerpy.connect(dsn="DBIP22024", user=group_name,
password=group_password)

    program(mydb)

    mydb.close()


if __name__ == "__main__":
    ssh_username = input("Enter your Studium username: ")
    ssh_password = getpass.getpass("Enter your Studium password A: ")

    tunnel = SSHTunnel(ssh_username, ssh_password, "groucho.it.uu.se",
22)
    tunnel.start(
        local_host="127.0.0.1",
        local_port=13600,
        remote_host="127.0.0.1",
        remote_port=1360,
    )

    # Now the tunnel is ready, connect to DB
    db_connect()

    # Stop the tunnel
```

```
tunnel.stop()
```

sshtunnel.py

```
import paramiko
import socket
import select
import threading
from multiprocessing import Process, Event

class SSHTunnel:
    """SSH Tunnel manager with start/stop interface"""

    def __init__(self, ssh_username, ssh_password, host, port):
        self.ssh_username = ssh_username
        self.ssh_password = ssh_password
        self.host = host
        self.port = port

        self.local_port = None
        self.local_host = None

        self.tunnel_opened = Event()
        self.close_tunnel = Event()
        self.client = None
        self.forward_thread = None

    def _pipe(self, src, dst):
        """Copy data both ways between sockets"""
        try:
            while True:
                r, w, x = select.select([src, dst], [], [])
                if src in r:
                    data = src.recv(1024)
                    if len(data) == 0:
                        break
                    dst.sendall(data)
                if dst in r:
                    data = dst.recv(1024)
                    if len(data) == 0:
                        break
                    src.sendall(data)
        finally:
            try:
                src.close()
            except EOFError:
```

```
                pass
            try:
                dst.close()
            except EOFError:
                pass

    def _forward_tunnel(self, local_host, local_port, remote_host,
remote_port):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.bind((local_host, local_port))
        self.local_host, self.local_port = sock.getsockname()
        sock.settimeout(1.0)
        sock.listen(2)

        self.tunnel_opened.set()

        while not self.close_tunnel.is_set():
            try:
                client, addr = sock.accept()
                chan = self.client.get_transport().open_channel(
                    "direct-tcpip",
                    (remote_host, remote_port),
                    client.getsockname()
                )
                threading.Thread(target=self._pipe, args=(client,
chan), daemon=True).start()
            except socket.timeout:
                continue

        sock.close()

    def _run_tunnel(self, local_host, local_port, remote_host,
remote_port):
        # Connect SSH
        self.client = paramiko.SSHClient()

self.client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        self.client.connect(
            hostname=self.host,
            port=self.port,
            username=self.ssh_username,
            password=self.ssh_password
        )
```

```
        # Start forwarding
        self._forward_tunnel(local_host, local_port, remote_host,
remote_port)

        # Clean up
        self.client.close()

    def start(self, local_host, local_port, remote_host, remote_port):
        """Start the SSH tunnel in a separate process"""
        self.process = Process(target=self._run_tunnel,
                               args=(local_host, local_port,
remote_host, remote_port))
        self.process.start()
        self.tunnel_opened.wait()  # Wait until tunnel is ready

    def stop(self):
        """Stop the tunnel"""
        self.close_tunnel.set()
        self.process.join()
```