Organization and Functioning of a Bookstore

Description of the Model. Rules of Operation

The data model will manage information related to the organization and operation of a bookstore. The bookstore operates in the online environment, based on orders placed by the bookstore's customers, who can be both individuals and legal entities. Once prepared by the bookstore employees, the orders are picked up by couriers and transported to the address specified by the customer. Since the bookstore works with several courier companies, the model also manages the information related to these companies.

Being a bookstore, the model also aims to organize the books it sells to customers. These are organized by departments according to the category they are in, facilitating a faster preparation of parcels. Library employees work in these departments. They know best how the books are organized within the assigned department, thus being much more efficient when preparing the orders.

At the same time, the model manages the information related to the authors of the books and the publishers where they were published. The author uses one or more publishers to publish his books.

Constraints

The data model respects the following operational restrictions:

→ An author cannot independently publish a book, but has to turn to a publishing house for this.
→ A book can be published in different editions by the same publisher.
→ A book can be published by several publishers, but it must be published by at least one.
→ An order must contain at least one book.
→ An order is transported by a single courier.
→ A book must be assigned to a single department.
→ A book can be part of only one category.
→ An employee can work in a single department.
→ A book is written by a single author.

Description of the Entities

For the data model related to the organization of a bookstore, the structures PUBLISHER, AUTHOR, BOOK, DEPARTMENT, EMPLOYEE, ORDER, COURIER, COURIER_COMPANY, CLIENT, INDIVIDUAL, COMPANY represent entities.

1. **AUTHOR**
    - An individual who collaborates with a publisher to publish their works.
    - Primary key: author_code.

2. **PUBLISHER**
    - A specialized company (legal entity) that deals with the translation, printing, publishing, and distribution of books to bookstores.
    - Primary key: publisher_id.
3. **BOOK**
    - Identifies the product sold by the bookstore, ordered by the customer, and delivered by the courier to the specified address.
    - Primary key: isbn.
4. **DEPARTMENT**
    - A subdivision of the bookstore where employees work, and where books of a certain category are stored and organized.
    - Primary key: department_code.
5. **EMPLOYEE**
    - An employee (individual, permanently employed) of the bookstore, responsible for organizing books within a department.
    - Primary key: personal_code.
6. **ORDER**
    - An order placed by the customer containing the books they wish to purchase.
    - Primary key: order_code.
7. **COURIER**
    - An individual, permanently employed in a courier company, responsible for transporting books ordered by the customer to the address specified by them.
    - Primary key: courier_id.
8. **COURIER_COMPANY**
    - A company (legal entity) specialized in the transportation of goods, employing couriers.
    - Primary key: company_id.
9. **CLIENT**
    - An individual or legal entity that purchases books from the bookstore's website.
    - Primary key: client_id.
10. **INDIVIDUAL**

- A subentity of the CLIENT entity that designates an individual purchasing books from the bookstore's website.

11. **COMPANY**
    - A subentity of the CLIENT entity that designates a company (legal entity) purchasing books from the bookstore's website.

Description of Relationships

Description of relationships, including specifying their cardinality.

The names of the connections are suggestive for the model, reflecting its content and the entities they link.

1. **AUTHOR_works_with_PUBLISHER**
    - A many-to-many relationship between the AUTHOR and PUBLISHER entities, reflecting the connection between them (authors collaborating with publishers to publish books).
    - Minimum cardinality: 1:1 (an author must work with at least one publisher to publish books, and a publisher must enter into at least one contract with an author to operate).
    - Maximum cardinality: m:n (an author can work with multiple publishers, and a publisher can work with multiple authors).
2. **PUBLISHER_publishes_BOOK**
    - A many-to-many relationship between the PUBLISHER and BOOK entities, reflecting the connection between them (books published by a publisher).
    - Minimum cardinality: 1:1 (a publisher must publish at least one book to operate, and a book must be published by at least one publisher).
    - Maximum cardinality: m:n (a publisher can publish any number of books, and a book can be published by multiple publishers).
3. **BOOK_belongs_to_DEPARTMENT**
    - The relationship between BOOK and DEPARTMENT reflects the connection between them (in which department a book has been assigned).
    - Minimum cardinality: 0:1 (there can be departments without books, but each book must be assigned to a department).
    - Maximum cardinality: n:1 (multiple books can belong to a department, but each book belongs to a unique department).
4. **EMPLOYEE_works_in_DEPARTMENT**

- The relationship between EMPLOYEE and DEPARTMENT reflects the connection between them (which employees work in each department).
- Minimum cardinality: 1:1 (at least one employee works in each department, and each employee works in only one department).
- Maximum cardinality: n:1 (multiple employees can work in a department, but each employee works in only one department).

5. **BOOK_is_part_of_ORDER**
    - A many-to-many relationship between the BOOK and ORDER entities, reflecting the connection between them (books ordered by the customer).
    - Minimum cardinality: 1:0 (each order must include at least one book, but not all books need to be part of an order).
    - Maximum cardinality: m:n (multiple books can be part of an order, and a book can be part of multiple orders).
6. **ORDER_is_handled_by_COURIER**
    - The relationship between ORDER and COURIER reflects the connection between them (which courier handles the transportation of an order).
    - Minimum cardinality: 1:1 (a courier transports at least one order, and an order is transported by a single courier).
    - Maximum cardinality: n:1 (a courier handles the transportation of multiple orders, and an order is transported by a single courier).
7. **COURIER_works_for_COURIER_COMPANY**
    - The relationship between COURIER and COURIER_COMPANY reflects the connection between them (which courier company each courier works for).
    - Minimum cardinality: 1:1 (each courier works for a single courier company, and a courier company employs at least one courier to operate).
    - Maximum cardinality: n:1 (multiple couriers can work for a courier company, but a courier works for only one courier company).
8. **CLIENT_places_ORDER**
    - The relationship between CLIENT and ORDER reflects the connection between them (orders placed by the bookstore's clients).
    - Minimum cardinality: 1:1 (to be considered a bookstore client, the client must place at least one order, and an order is placed by a single client).
    - Maximum cardinality: 1:n (a client can place multiple orders, but an order is placed by a single client).

Description of Attributes

Description of attributes, including data type and any constraints, default values, and possible values for the attributes.

The entity **AUTHOR** has the following attributes:

- author_id = integer variable, maximum length 10, representing the code associated with the author in the database.
- last_name = character variable, maximum length 100, representing the last name of the author.
- first_name = character variable, maximum length 100, representing the first name of the author.

The entity **PUBLISHER** has the following attributes:

- publisher_id = integer variable, maximum length 10, representing the code associated with the publisher in the database.
- publisher_name = character variable, maximum length 100, representing the name of the publisher.
- phone_number = character variable, maximum length 20, representing the phone number at which the publisher can be contacted.
- email = character variable, maximum length 100, representing the email address at which the publisher can be contacted.

The entity **BOOK** has the following attributes:

- isbn = character variable, maximum length 20, representing the ISBN of the book.
- author_id = integer variable, maximum length 10, representing the code associated with the author who wrote the book in the database. This attribute corresponds to the primary key value of the AUTHOR table.
- department_code = integer variable, maximum length 10, representing the code associated with the department to which the book is assigned in the database. This attribute corresponds to the primary key value of the DEPARTMENT table.
- title = character variable, maximum length 100, representing the title of the book.
- category = character variable, maximum length 100, representing the category to which a book belongs.
- publication_date = calendar date variable, representing the date the book was first published.

- number_of_copies = integer variable, maximum length 10, representing the number of copies of the book available for sale.
- price = integer variable, maximum length, representing the price at which the book is sold.

The entity **DEPARTMENT** has the following attributes:

- department_code = integer variable, maximum length 10, representing the code associated with each department in the database.
- department_name = character variable, maximum length 100, representing the name of the department to which a book belongs.
- number_of_employees = integer variable, maximum length 10, representing the number of employees working in each department.

The entity **EMPLOYEE** has the following attributes:

- personal_code = character variable, maximum length 20, representing the personal identification code of the employee.
- department_code = integer variable, maximum length 10, representing the code associated with the department in which the employee works in the database. This attribute corresponds to the primary key value of the DEPARTMENT table.
- salary = integer variable, maximum length 10, representing the net monthly salary of the employee.
- commission = integer variable, maximum length 10, with a default value of null, representing the commission earned by the employee from sales, expressed in percentages.
- last_name = character variable, maximum length 100, representing the last name of the employee.
- first_name = character variable, maximum length 100, representing the first name of the employee.
- phone_number = character variable, maximum length 20, representing the phone number at which the employee can be contacted.
- email = character variable, maximum length 100, representing the email address at which the employee can be contacted.
- address = character variable, maximum length 100, representing the address where the employee lives.

The entity **ORDER** has the following attributes:

- order_code = integer variable, maximum length 10, representing the code associated with the order in the database.
- courier_id = integer variable, maximum length 10, representing the code associated with the courier transporting the order in the database. This attribute corresponds to the primary key value of the COURIER table.
- client_id = integer variable, maximum length 10, representing the code associated with the client who placed the order in the database. This attribute corresponds to the primary key value of the CLIENT table.
- payment_method = character variable, maximum length 20, representing the payment method chosen by the client. It can be either 'cash on delivery' when receiving the package or 'online' when the client chooses to pay with a card online.
- number_of_products = integer variable, maximum length 10, representing the number of products ordered by a client.

The entity **COURIER** has the following attributes:

- courier_id = integer variable, maximum length 10, representing the code associated with the courier in the database.
- company_id = integer variable, maximum length 10, representing the code associated with the courier company for which the courier works in the database. This attribute corresponds to the primary key value of the COURIER_COMPANY table.
- last_name = character variable, maximum length 100, representing the last name of the courier.
- first_name = character variable, maximum length 100, representing the first name of the courier.
- phone_number = character variable, maximum length 20, representing the phone number at which the courier can be contacted.
- email = character variable, maximum length 100, representing the email address at which the courier can be contacted.

The entity **COURIER_COMPANY** has the following attributes:

- company_id = integer variable, maximum length 10, representing the code associated with the company in the database.
- company_name = character variable, maximum length 100, representing the name of the courier company.
- phone_number = character variable, maximum length 20, representing the phone number at which the courier company can be contacted.

- email = character variable, maximum length 100, representing the email address at which the courier company can be contacted.

The entity **CLIENT** has the following attributes:

- client_id = integer variable, maximum length 10, representing the code associated with the client in the database.
- phone_number = character variable, maximum length 20, representing the phone number at which the client can be contacted.
- email = character variable, maximum length 100, representing the email address at which the client can be contacted.
- address = character variable, maximum length 100, representing the address to which the order will be delivered.

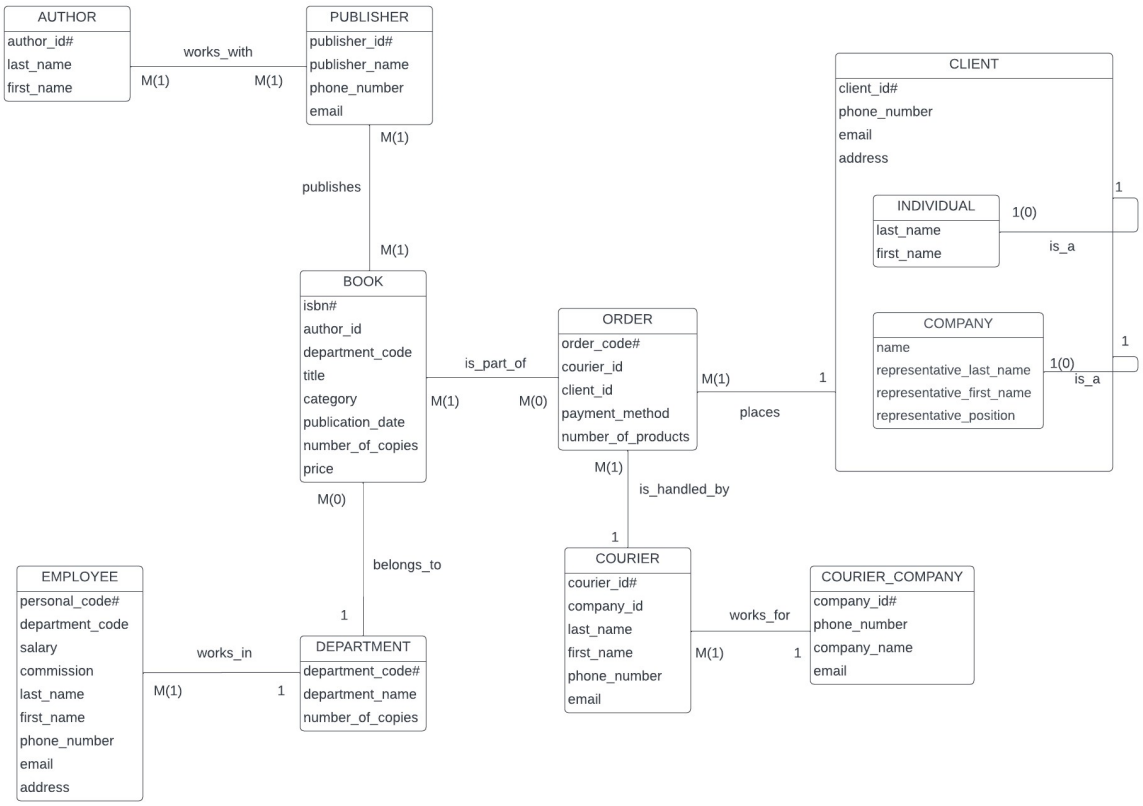
The entity **INDIVIDUAL** has the following attributes:

- client_id = integer variable, maximum length 10, representing the code associated with the client in the database.
- last_name = character variable, maximum length 100, representing the last name of the client.
- first_name = character variable, maximum length 100, representing the first name of the client.

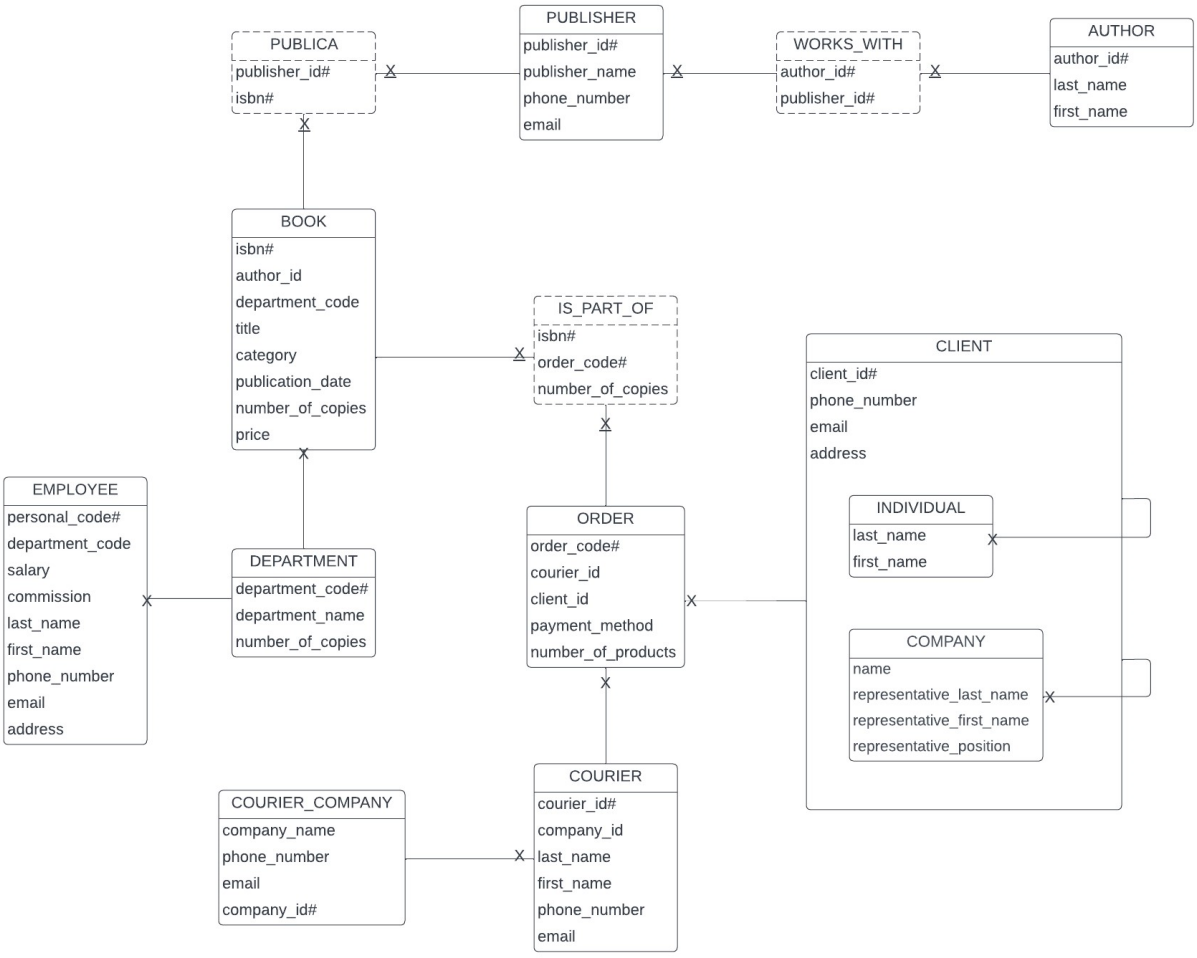
The entity **COMPANY** has the following attributes:

- client_id = integer variable, maximum length 10, representing the code associated with the client in the database.
- name = character variable, maximum length 100, representing the name of the company.
- representative_last_name = character variable, maximum length 100, representing the last name of the company representative.
- representative_first_name = character variable, maximum length 100, representing the first name of the company representative.
- representative_position = character variable, maximum length 100, representing the position of the company representative.

The Entity-Relationship Diagram



The Conceptual Diagram



The Relational Schemas

Listing the relational schemas corresponding to the designed conceptual diagram at point 7.

1. **AUTHOR** (`author_code#` , `last_name` , `first_name`)
2. **WORKS_WITH** (`publisher_id#` , `author_code#`)
3. **PUBLISHER** (`publisher_id#` , `publisher_name` , `phone_number` , `email`)
4. **PUBLISHES** (`publisher_id#` , `isbn#`)
5. **BOOK** (`isbn#` , `author_code` , `department_code` , `title` , `category` , `publication_date` , `number_of_copies` , `price`)
6. **DEPARTMENT** (`department_code#` , `department_name` , `number_of_employees`)

7. **EMPLOYEE** (`personal_code#`, `department_code`, `salary`, `commission`, `last_name`, `first_name`, `phone_number`, `email`, `address`)
8. **IS_PART_OF** (`order_code#`, `isbn#`)
9. **ORDER** (`order_code#`, `courier_id`, `client_id`, `payment_method`, `number_of_products`)
10. **COURIER** (`courier_id#`, `company_id`, `last_name`, `first_name`, `phone_number`, `email`)
11. **WORKS_FOR** (`courier_id#`, `company_id#`)
12. **COURIER_COMPANY** (`company_id#`, `company_name`, `phone_number`, `email`)
13. **CLIENT** (`client_id#`, `phone_number`, `email`, `address`)
14. **INDIVIDUAL** (`client_id#`, `client_last_name`, `client_first_name`)
15. **COMPANY** (`client_id#`, `company_name`, `representative_last_name`, `representative_first_name`, `representative_position`)

The Normalization Process

1. **First Normal Form (1NF):**
   - Eliminate duplicate columns from the same table.
   - Create a separate table for each group of related data and identify each row with a unique column or set of columns (the primary key).
2. **Second Normal Form (2NF):**
   - Meet the requirements of 1NF.
   - Remove subsets of data that apply to multiple rows and place them in separate tables.
   - Create relationships between these new tables and their predecessors through the use of foreign keys.
3. **Third Normal Form (3NF):**
   - Meet the requirements of 2NF.
   - Remove columns that are not dependent upon the primary key.
   - Non-prime attributes (attributes not part of the primary key) should not depend on other non-prime attributes.

Normalization process up to the third normal form (FN1-FN3) has been carried out.

Given the initial relational schemas:

AUTHOR (author_code#, last_name, first_name, **publisher_name**)
PUBLISHES (publisher_id#, isbn#)
BOOK (isbn#, author_id, department_code, title, category, publication_date, number_of_copies, price)
DEPARTMENT (department_code#, department_name, number_of_employees)
EMPLOYEE (personal_code#, department_code, salary, commission, last_name, first_name, phone_number, email, address)
IS_PART_OF (order_code#, isbn#, client_id, payment_method, number_of_products)
COURIER(courier_id#, company_id, company_name, last_name, first_name, phone_number, email)
CLIENT (client_id#, phone_number, email, address)
INDIVIDUAL (last_name, first_name)
COMPANY (company_name, representative_last_name, representative_first_name, representative_position)

Non-1NF

The AUTHOR table violates 1NF as it contains the publisher_name field, which corresponds to the name of the publishing house. Since an author can work with multiple publishers, the attribute publisher_name does not correspond to an indivisible value. To bring the schema into 1NF, I create the PUBLISHER table, which contains relevant information, and the associative table WORKS_WITH.

Revised relational schemas:

AUTHOR (author_code#, last_name, first_name)
WORKS_WITH(publisher_id#, author_code#)
PUBLISHER (publisher_id#, publisher_name, phone_number, email)
PUBLISHES (publisher_id#, isbn#)
BOOK (isbn#, author_code, department_code, title, category, publication_date, number_of_copies, price)
DEPARTMENT (department_code#, department_name, number_of_employees)
EMPLOYEE (cnp#, department_code, salary, commission, last_name, first_name, phone_number, email, address)
IS_PART_OF (order_code#, isbn#, **id_client, payment_method, number_of_products**)
COURIER (courier_id#, company_id, company_name, last_name, first_name, phone_number, email)
CLIENT (client_id#, phone_number, email, address)
INDIVIDUAL (client_id, client_last_name, client_first_name)
COMPANY (client_id, company_name, representative_last_name, representative_first_name, representative_position)

Non-2NF

Continuing, it is observed that the diagram is not in 2NF because in the IS_PART_OF table, the attributes client_id, payment_method, and number_of_products depend only on the primary key order_code and not on the primary key isbn. Therefore, they depend only partially on the primary key.

To bring the diagram to 2NF, I will need to create the ORDER table, which will contain the information for the id_client, payment_method, and number_of_products fields.

After the specified changes, the relational schemes are as follows:

AUTHOR (author_code#, last_name, first_name)
WORKS_WITH(publisher_id#, author_code#)
PUBLISHER (publisher_id#, publisher_name, phone_number, email)
PUBLISHES(publisher_id#, isbn#)
BOOK (isbn#, author_code, department_code, title, category, publication_date, number_of_copies, price)
DEPARTMENT (department_code#, department_name, number_of_employees)
EMPLOYEE (personal_code#, department_code, salary, commission, last_name, first_name, phone_number, email, address)
PART_OF_ORDER(order_code#, #isbn)
ORDER(order_code#, courier_id, client_id, payment_method, number_of_products)
COURIER (courier_id#, **company_id, company_name**, last_name, first_name, phone_number, email)
CLIENT (client_id#, phone_number, email, address)
INDIVIDUAL (client_id#, client_last_name, client_first_name)
COMPANY (client_id#, company_name, representative_last_name, representative_first_name, representative_position)

Non-3NF

In the end, it is observed that the diagram is not in FN3 because in the COURIER table, the company_name depends on company_id, creating transitive dependencies. To bring the diagram to FN3, I will create a new table, COURIER_COMPANY, which will contain relevant information for this.

In the end, the relational schemes will be as follows:

AUTHOR (author_code#, last_name, first_name)
WORKS_WITH(publisher_id#, author_code#)
PUBLISHER (publisher_id#, publisher_name, phone_number, email)
PUBLISHES(publisher_id#, isbn#)
BOOK (isbn#, author_code, department_code, title, category, publication_date, number_of_copies, price)
DEPARTMENT (department_code#, department_name, number_of_employees)
EMPLOYEE (personal_code#, department_code, salary, commission, last_name, first_name, phone_number, email, address)
PART_OF_ORDER(order_code#, isbn#)
ORDER(order_code#, courier_id, client_id, payment_method, number_of_products)
COURIER (courier_id#, company_id, last_name, first_name, phone_number, email)
COURIER_COMPANY (company_id#, company_name, phone_number, email)

CLIENT (client_id#, phone_number, email, address)
INDIVIDUAL (client_id#, client_last_name, client_first_name)
COMPANY (client_id#, company_name, representative_last_name, representative_first_name, representative_position)

Sequence for Inserting Records into Tables

```
CREATE SEQUENCE record_sequence
MINVALUE 1
MAXVALUE 1000000
START WITH 1
INCREMENT BY 1
NOCYCLE;
```

Creating Tables and Inserting Sample Data

Author Table

AUTHOR_ID	LAST_NAME	FIRST_NAME
1	Christie	Agatha
2	King	Stephen
3	Asimov	Isaac
4	Tolkien	J. R. R.
5	Defoe	Daniel
6	Backman	Fredrik
7	Alex	Michaleides
8	Hawking	Stephen
9	Haig	Matt

Table Creation

```
create table author
(
    author_id int not null primary key,
    last_name varchar2(100) not null,
    first_name varchar2(100) not null
);
```

Sample Data

```
INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Christie', 'Agatha'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'King', 'Stephen'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Asimov', 'Isaac'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Tolkien', 'J. R. R.'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Defoe', 'Daniel'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Backman', 'Fredrik'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Alex', 'Michaleides'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Hawking', 'Stephen'
);

INSERT INTO author(author_id, last_name, first_name)
VALUES
(
    record_sequence.nextval, 'Haig', 'Matt'
);
```

Works_With TABLE

PUBLISHER_ID	AUTHOR_ID
1	3
1	6
2	4
2	5
3	2
3	3
3	9
4	1
4	7
5	8

Table Creation

```
CREATE TABLE works_with
(
    publisher_id int NOT NULL,
    author_id int NOT NULL,
    PRIMARY KEY (publisher_id, author_id)
);
```

Sample data

```
INSERT INTO works_with(publisher_id, author_id)
VALUES
(1, 3);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(1, 6);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(2, 5);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(2, 4);

INSERT INTO works_with(publisher_id, author_id)
VALUES
```

```
(3, 9);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(3, 2);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(3, 3);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(4, 1);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(5, 8);

INSERT INTO works_with(publisher_id, author_id)
VALUES
(4, 7);
```

Publisher Table

PUBLISHER_ID	PUBLISHER_NAME	PHONE_NUMBER	EMAIL
1	Penguin Books	123-456-7890	penguin@example.com
2	HarperCollins	987-654-3210	harper@example.com
3	Random House	111-222-3333	randomhouse@example.com
4	Simon & Schuster	555-666-7777	simon@example.com
5	Hachette Book Group	999-888-7777	hachette@example.com

Table Creation

```
create table publisher
(
    publisher_id int not null primary key,
    publisher_name varchar2(100) not null,
    phone_number varchar2(20) not null,
    email varchar2(100) not null
);
```

Sample Data

```
INSERT INTO publisher (publisher_id, publisher_name, phone_number, email)
VALUES
(1, 'Penguin Books', '123-456-7890', 'penguin@example.com');

INSERT INTO publisher (publisher_id, publisher_name, phone_number, email)
VALUES
(2, 'HarperCollins', '987-654-3210', 'harper@example.com');

INSERT INTO publisher (publisher_id, publisher_name, phone_number, email)
VALUES
(3, 'Random House', '111-222-3333', 'randomhouse@example.com');

INSERT INTO publisher (publisher_id, publisher_name, phone_number, email)
VALUES
(4, 'Simon & Schuster', '555-666-7777', 'simon@example.com');

INSERT INTO publisher (publisher_id, publisher_name, phone_number, email)
VALUES
(5, 'Hachette Book Group', '999-888-7777', 'hachette@example.com');
```

Publishes Table

PUBLISHER_ID	ISBN
1	9780385121675
1	9780451112241
2	9780525520914
2	9780553293357
2	9781476738024
3	9780553109535
3	9780553380163
3	9780618989344
4	9781250301697
4	9781420958145
5	9781476738024

Table Creation

```
CREATE TABLE publishes
(
    publisher_id int NOT NULL,
    isbn int NOT NULL,
    PRIMARY KEY (publisher_id, isbn)
);
```

Sample Data

```
INSERT INTO publishes(publisher_id, isbn)
VALUES
(1, '9780007113804');

INSERT INTO publishes(publisher_id, isbn)
VALUES
(1, '9780670813029');

INSERT INTO publishes(publisher_id, isbn)
VALUES
(2, '9780553294385');

INSERT INTO publishes(publisher_id, isbn)
VALUES
(3, '9780345339706');


INSERT INTO publishes(publisher_id, isbn)
VALUES
(5, '9780553380163');

INSERT INTO publishes(publisher_id, isbn)
VALUES
(4, '9781476738024');
```

Department Table

DEPARTMENT_CODE	DEPARTMENT_NAME	NUMBER_OF_EMPLOYEES
1	Mystery/Thriller	10
2	Horror	9
3	Science Fiction	11
4	Adventure	8
5	Contemporary Fiction	15
6	Science/Popular Science	7

Table Creation

```
CREATE TABLE department
(
    department_code int not null primary key,
    department_name varchar2(100) not null,
    number_of_employees int not null
);
```

Sample Data

```
INSERT INTO department (department_code, department_name, number_of_employees)
VALUES
(1, 'Mystery/Thriller', 10);

INSERT INTO department (department_code, department_name, number_of_employees)
VALUES
(2, 'Horror', 9);

INSERT INTO department (department_code, department_name, number_of_employees)
VALUES
(3, 'Science Fiction', 11);

INSERT INTO department (department_code, department_name, number_of_employees)
VALUES
(4, 'Adventure', 8);

INSERT INTO department (department_code, department_name, number_of_employees)
VALUES
(5, 'Contemporary Fiction', 15);


INSERT INTO department (department_code, department_name, number_of_employees)
VALUES
(6, 'Science/Popular Science', 7);
```

Book Table

ISBN	AUTHOR_ID	DEPARTMENT_CODE	TITLE	CATEGORY	PUBLICATION_DATE	NUMBER_OF_COPIES	PRICE
9780553380163	8	6	A Brief History of Time	Science	01-SEP-98	50	40
9780007113804	1	1	The Murder of Roger Ackroyd	Mystery/Thriller	10-JUN-02	100	25
9780670813029	2	2	It	Horror	15-SEP-86	150	30
9780553294385	3	3	Foundation	Science Fiction	01-JUN-91	120	20
9780345339706	4	4	The Hobbit	Adventure	12-JUL-86	200	35
9780486264671	5	4	Robinson Crusoe	Adventure	10-NOV-97	80	15
9781476738024	6	5	A Man Called Ove	Fiction	15-JUL-14	75	28

Table Creation

```
CREATE TABLE book
(
    isbn varchar2(20) not null primary key,
    author_id int not null,
    department_code int not null,
```

```
    title varchar2(100) not null,
    category varchar2(100) not null,
    publication_date date not null,
    number_of_copies int not null,
    price int not null,
    CONSTRAINT fk_book_author
        FOREIGN KEY (author_id)
        REFERENCES author(author_id),
    CONSTRAINT fk_book_department
        FOREIGN KEY (department_code)
        REFERENCES department(department_code)
);
```

Sample Data

```
-- Agatha Christie - Mystery/Thriller
INSERT INTO book (isbn, author_id, department_code, title, category,
 publication_date, number_of_copies, price)
VALUES
('9780007113804', 1, 1, 'The Murder of Roger Ackroyd', 'Mystery/Thriller',
TO_DATE('2002-06-10', 'YYYY-MM-DD'), 100, 25);

-- Stephen King - Horror
INSERT INTO book (isbn, author_id, department_code, title, category,
publication_date, number_of_copies, price)
VALUES
('9780670813029', 2, 2, 'It', 'Horror',
TO_DATE('1986-09-15', 'YYYY-MM-DD'), 150, 30);

-- Isaac Asimov - Science Fiction
INSERT INTO book (isbn, author_id, department_code, title, category,
 publication_date, number_of_copies, price)
VALUES
('9780553294385', 3, 3, 'Foundation', 'Science Fiction',
TO_DATE('1991-06-01', 'YYYY-MM-DD'), 120, 20);

-- J.R.R. Tolkien - Adventure
INSERT INTO book (isbn, author_id, department_code, title, category,
publication_date, number_of_copies, price)
VALUES
('9780345339706', 4, 4, 'The Hobbit', 'Adventure',
TO_DATE('1986-07-12', 'YYYY-MM-DD'), 200, 35);

-- Daniel Defoe - Adventure
INSERT INTO book (isbn, author_id, department_code, title, category,
publication_date, number_of_copies, price)
VALUES
('9780486264671', 5, 4, 'Robinson Crusoe', 'Adventure',
TO_DATE('1997-11-10', 'YYYY-MM-DD'), 80, 15);

-- Stephen Hawking - Science
INSERT INTO book (isbn, author_id, department_code, title, category,
publication_date, number_of_copies, price)
VALUES
('9780553380163', 8, 6, 'A Brief History of Time', 'Science',
```

```
TO_DATE('1998-09-01', 'YYYY-MM-DD'), 50, 40);

-- Fredrik Backman - Fiction
INSERT INTO book (isbn, author_id, department_code, title, category,
publication_date, number_of_copies, price)
VALUES
('9781476738024', 6, 5, 'A Man Called Ove', 'Fiction',
TO_DATE('2014-07-15', 'YYYY-MM-DD'), 75, 28);
```

Employee Table

PERSONAL_CODE	DEPARTMENT_CODE	SALARY	COMMISSION	LAST_NAME	FIRST_NAME	PHONE_NUMBER	EMAIL	ADDRESS
123456789	1	50000	-	Smith	John	123-456-7890	john.smith@email.com	123 Main St
987654321	2	60000	5	Johnson	Alice	987-654-3210	alice.johnson@email.com	456 Oak St
456789012	3	55000	3	Davis	Robert	456-789-0123	robert.davis@email.com	789 Pine St
789012345	1	52000	2	Miller	Emily	789-012-3456	emily.miller@email.com	234 Elm St
567890123	2	58000	-	Wilson	David	567-890-1234	david.wilson@email.com	567 Birch St
901234567	3	53000	4	Brown	Sophia	901-234-5678	sophia.brown@email.com	890 Maple St
234567890	1	51000	-	Anderson	Michael	234-567-8901	michael.anderson@email.com	678 Cedar St
345678901	2	59000	6	Davis	Olivia	345-678-9012	olivia.davis@email.com	901 Oak St

Table Creation

```
CREATE TABLE employee
(
    personal_code varchar2(20) not null primary key,
    department_code int not null,
    salary int not null,
    commission int default null,
    last_name varchar2(100) not null,
    first_name varchar2(100) not null,
    phone_number varchar2(20) not null,
    email varchar2(100) not null,
    address varchar2(100) not null,
    CONSTRAINT fk_employee_department
        FOREIGN KEY (department_code)
        REFERENCES department(department_code)
);
```

Sample Data

```
INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
```

```
VALUES
('123456789', 1, 50000, NULL, 'Smith', 'John', '123-456-7890',
   'john.smith@email.com', '123 Main St');

INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
VALUES ('987654321', 2, 60000, 5, 'Johnson', 'Alice', '987-654-3210',
  'alice.johnson@email.com', '456 Oak St');

INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
VALUES ('456789012', 3, 55000, 3, 'Davis', 'Robert', '456-789-0123',
  'robert.davis@email.com', '789 Pine St');

INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
VALUES ('789012345', 1, 52000, 2, 'Miller', 'Emily', '789-012-3456',
  'emily.miller@email.com', '234 Elm St');

INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
VALUES ('567890123', 2, 58000, NULL, 'Wilson', 'David', '567-890-1234',
 'david.wilson@email.com', '567 Birch St');

INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
VALUES ('901234567', 3, 53000, 4, 'Brown', 'Sophia', '901-234-5678',
  'sophia.brown@email.com', '890 Maple St');

INSERT INTO employee(personal_code, department_code, salary, commission,
   last_name, first_name, phone_number, email, address)
VALUES ('234567890', 1, 51000, NULL, 'Anderson', 'Michael', '234-567-8901',
  'michael.anderson@email.com', '678 Cedar St');

INSERT INTO employee(personal_code, department_code, salary, commission,
  last_name, first_name, phone_number, email, address)
VALUES ('345678901', 2, 59000, 6, 'Davis', 'Olivia', '345-678-9012',
  'olivia.davis@email.com', '901 Oak St');
```

Courier_Company Table

COMPANY_ID	COMPANY_NAME	PHONE_NUMBER	EMAIL
1	FedEx	123-456-7890	info@fedex.com
2	UPS	987-654-3210	info@ups.com
3	DHL	555-123-4567	info@dhl.com
4	USPS	888-999-0000	info@usps.com
5	TNT Express	333-555-7777	info@tntexpress.com

Table Creation

```
CREATE TABLE courier_company
(
    company_id int NOT NULL PRIMARY KEY,
    company_name varchar2(100) NOT NULL,
    phone_number varchar(20) NOT NULL,
    email varchar2(100) NOT NULL
);
```

Sample Data

```
INSERT INTO courier_company (company_id, company_name, phone_number, email)
VALUES
(1, 'FedEx', '123-456-7890', 'info@fedex.com');

INSERT INTO courier_company (company_id, company_name, phone_number, email)
VALUES
(2, 'UPS', '987-654-3210', 'info@ups.com');

INSERT INTO courier_company (company_id, company_name, phone_number, email)
VALUES
(3, 'DHL', '555-123-4567', 'info@dhl.com');

INSERT INTO courier_company (company_id, company_name, phone_number, email)
VALUES
(4, 'USPS', '888-999-0000', 'info@usps.com');

INSERT INTO courier_company (company_id, company_name, phone_number, email)
VALUES
(5, 'TNT Express', '333-555-7777', 'info@tntexpress.com');
```

Courier Table

COURIER_ID	COMPANY_ID	LAST_NAME	FIRST_NAME	PHONE_NUMBER	EMAIL
1	1	Smith	John	123-456-7890	john.smith@example.com
2	2	Johnson	Emma	987-654-3210	emma.johnson@example.com
3	3	Williams	Michael	555-123-4567	michael.williams@example.com
4	1	Davis	Sophia	111-222-3333	sophia.davis@example.com
5	2	Anderson	Daniel	999-888-7777	daniel.anderson@example.com

Table Creation

```
CREATE TABLE courier
(
```

```
    courier_id int not null primary key,
    company_id int not null,
    last_name varchar2(100) not null,
    first_name varchar2(100) not null,
    phone_number varchar2(20) not null,
    email varchar2(100) not null,
    CONSTRAINT fk_courier_company
        FOREIGN KEY (company_id)
        REFERENCES courier_company(company_id)
);
```

Sample Data

```
INSERT INTO courier (courier_id, company_id, last_name, first_name, phone_number, email)
VALUES
(1, 1, 'Smith', 'John', '123-456-7890', 'john.smith@example.com');

INSERT INTO courier (courier_id, company_id, last_name, first_name, phone_number, email)
VALUES
(2, 2, 'Johnson', 'Emma', '987-654-3210', 'emma.johnson@example.com');

INSERT INTO courier (courier_id, company_id, last_name, first_name, phone_number, email)
VALUES
(3, 3, 'Williams', 'Michael', '555-123-4567', 'michael.williams@example.com');

INSERT INTO courier (courier_id, company_id, last_name, first_name, phone_number, email)
VALUES
(4, 1, 'Davis', 'Sophia', '111-222-3333', 'sophia.davis@example.com');

INSERT INTO courier (courier_id, company_id, last_name, first_name, phone_number, email)
VALUES
(5, 2, 'Anderson', 'Daniel', '999-888-7777', 'daniel.anderson@example.com');
```

Client Table

CLIENT_ID	PHONE_NUMBER	EMAIL	ADDRESS
1	555-1234	contact@realbooklight.com	123 Main St
2	555-5678	contact@clearglow.com	456 Oak St
3	555-9101	contact@atutobookcase.com	789 Pine St
4	555-2345	mary-smith@example.com	101 Elm St
5	555-6789	john-doe@example.com	202 Birch St
6	555-1122	emily-jones@example.com	303 Maple St
7	555-3344	contact@novelhaven.com	404 Cedar St
8	555-5566	contact@fictionemporium.com	505 Walnut St
9	555-7788	contact@whimsicalreads.com	606 Spruce St
10	555-9900	contact@dreamybooks.com	707 Ash St
11	555-1122	noah-wilson@example.com	808 Oak St
12	555-3344	sophia-miller@example.com	909 Pine St
13	555-5566	jackson-jones@example.com	1010 Birch St
14	555-7788	emma-martin@example.com	1111 Maple St
15	555-9900	aiden-thomas@example.com	1212 Cedar St
16	555-1122	olivia-jackson@example.com	1313 Walnut St
17	555-3344	liam-brown@example.com	1414 Spruce St
18	555-1111	contact@bookmagic.com	808 Elm St
19	555-2222	contact@enchantingbooks.com	909 Oak St
20	555-3333	contact@imaginativepress.com	101 Pine St

Table Creation

```
CREATE TABLE client
(
    client_id int not null primary key,
    phone_number varchar2(20) not null,
    email varchar2(100) not null,
    address varchar2(100) not null
);
```

Sample Data

```
INSERT INTO client (client_id, phone_number, email, address)
VALUES
(1, '555-1234', 'contact@realbooklight.com', '123 Main St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(2, '555-5678', 'contact@clearglow.com', '456 Oak St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(3, '555-9101', 'contact@atutobookcase.com', '789 Pine St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(4, '555-2345', 'mary-smith@example.com', '101 Elm St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(5, '555-6789', 'john-doe@example.com', '202 Birch St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(6, '555-1122', 'emily-jones@example.com', '303 Maple St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(7, '555-3344', 'contact@novelhaven.com', '404 Cedar St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(8, '555-5566', 'contact@fictionemporium.com', '505 Walnut St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(9, '555-7788', 'contact@whimsicalreads.com', '606 Spruce St');

INSERT INTO client (id_client, numar_telefon, email, adresa)
VALUES
(10, '555-9900', 'contact@dreamybooks.com', '707 Ash St');

INSERT INTO client (client_id, phone_number, email, address)
```

```
VALUES
(11, '555-1122', 'noah-wilson@example.com', '808 Oak St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(12, '555-3344', 'sophia-miller@example.com', '909 Pine St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(13, '555-5566', 'jackson-jones@example.com', '1010 Birch St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(14, '555-7788', 'emma-martin@example.com', '1111 Maple St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(15, '555-9900', 'aiden-thomas@example.com', '1212 Cedar St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(16, '555-1122', 'olivia-jackson@example.com', '1313 Walnut St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(17, '555-3344', 'liam-brown@example.com', '1414 Spruce St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(18, '555-1111', 'contact@bookmagic.com', '808 Elm St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(19, '555-2222', 'contact@enchantingbooks.com', '909 Oak St');

INSERT INTO client (client_id, phone_number, email, address)
VALUES
(20, '555-3333', 'contact@imaginativepress.com', '101 Pine St');
```

Individual Table

CLIENT_ID	LAST_NAME	FIRST_NAME
4	Smith	Mary
5	Doe	John
6	Jones	Emily
11	Wilson	Noah
12	Miller	Sophia
13	Jones	Jackson
14	Martin	Emma
15	Thomas	Aiden
16	Jackson	Olivia
17	Brown	Liam

Table Creation

```
create table individual
(
client_id int not null primary key,
last_name varchar2(100) not null,
first_name varchar2(100) not null
);
```

Sample Data

```
INSERT INTO individual(client_id, last_name, first_name)
VALUES
(4, 'Smith', 'Mary');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(5, 'Doe', 'John');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(6, 'Jones', 'Emily');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(11, 'Wilson', 'Noah');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(12, 'Miller', 'Sophia');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
```

```
(13, 'Jones', 'Jackson');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(14, 'Martin', 'Emma');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(15, 'Thomas', 'Aiden');

INSERT INTO  individual(client_id, last_name, first_name)
VALUES
(16, 'Jackson', 'Olivia');

INSERT INTO individual(client_id, last_name, first_name)
VALUES
(17, 'Brown', 'Liam');
```

Company Table

CLIENT_ID	NAME	REPRESENTATIVE_LAST_NAME	REPRESENTATIVE_FIRST_NAME	REPRESENTATIVE_FUNCTION
1	RealBookLight	Rodriguez	Elena	CEO
2	ClearGlow	Mendez	Carlos	Manager
3	AutoBookcase	Garcia	Sofia	Director
7	NovelHeaven	Martinez	Javier	President
8	FictionEmporium	Turner	Liam	Manager
9	WhimsicalReads	Walker	Ethan	CEO
10	DreamyBooks	Thomas	Aiden	Manager
18	BookMagic	Brown	Liam	Director
19	EnchantingBooks	Wilson	Noah	CEO
20	ImaginativePress	Jackson	Olivia	Manager

Table Creation

```
CREATE TABLE company
(
    client_id int not null primary key,
    name varchar2(100) not null,
    representative_last_name varchar2(100) not null,
    representative_first_name varchar2(100) not null,
    representative_function varchar2(100) not null
);
```

Sample Data

```
INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(1, 'RealBookLight', 'Rodriguez', 'Elena', 'CEO');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(2, 'ClearGlow', 'Mendez', 'Carlos', 'Manager');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(3, 'AutoBookcase', 'Garcia', 'Sofia', 'Director');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(7, 'NovelHeaven', 'Martinez', 'Javier', 'President');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(8, 'FictionEmporium', 'Turner', 'Liam', 'Manager');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(9, 'WhimsicalReads', 'Walker', 'Ethan', 'CEO');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(10, 'DreamyBooks', 'Thomas', 'Aiden', 'Manager');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(18, 'BookMagic', 'Brown', 'Liam', 'Director');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(19, 'EnchantingBooks', 'Wilson', 'Noah', 'CEO');

INSERT INTO company (client_id, name, representative_last_name,
  representative_first_name, representative_function)
VALUES
(20, 'ImaginativePress', 'Jackson', 'Olivia', 'Manager');
```

Order Table

- I named this table client_order, instead of just order, because order is reserved as a key word in SQL

ORDER_CODE	COURIER_ID	CLIENT_ID	PAYMENT_METHOD	NUMBER_OF_PRODUCTS
1	1	4	Credit Card	3
2	2	5	Cash	2
3	3	6	PayPal	1
4	4	7	Credit Card	4
5	5	8	Cash	2
6	1	11	Credit Card	3
7	2	12	Cash	2
8	3	13	PayPal	1
9	4	14	Credit Card	4
10	5	15	Cash	2

Table Creation

```
CREATE TABLE client_order
(
    order_code int not null primary key,
    courier_id int not null,
    client_id int not null,
    payment_method varchar2(10) not null,
    number_of_products int not null,
    CONSTRAINT fk_client_order_courier
        FOREIGN KEY (courier_id)
        REFERENCES courier (courier_id),
    CONSTRAINT fk_client_order_client
        FOREIGN KEY (client_id)
        REFERENCES client (client_id)
);
```

Sample Data

```
INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (1, 1, 4, 'Credit Card', 3);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (2, 2, 5, 'Cash', 2);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (3, 3, 6, 'PayPal', 1);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
```

```
  number_of_products)
VALUES
    (4, 4, 7, 'Credit Card', 4);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
   number_of_products)
VALUES
    (5, 5, 8, 'Cash', 2);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (6, 1, 11, 'Credit Card', 3);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (7, 2, 12, 'Cash', 2);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (8, 3, 13, 'PayPal', 1);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (9, 4, 14, 'Credit Card', 4);

INSERT INTO client_order (order_code, courier_id, client_id, payment_method,
  number_of_products)
VALUES
    (10, 5, 15, 'Cash', 2);
```

Is_Part_Of Table

Table Creation

ISBN	ORDER_CODE	NUMBER_OF_COPIES
9780007113804	1	2
9780670813029	1	1
9780670813029	2	2
9780553294385	3	1
9780345339706	4	1
9780553294385	4	1
9780670813029	4	1
9780486264671	4	1
9780486264671	5	2
9780007113804	6	1
9780670813029	6	1
9780486264671	6	1
9780670813029	7	2
9780553294385	8	1
9780345339706	9	4
9780486264671	10	2

```
create table is_part_of
(
    isbn int not null,
    order_code int not null,
    number_of_copies int not null,
    primary key(isbn, order_code)
);
```

Sample Data

```
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780007113804', 1, 2);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780670813029', 1, 1);
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780670813029', 2, 2);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780553294385', 3, 1);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780345339706', 4, 1);
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780553294385', 4, 1);
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780670813029', 4, 1);
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
```

```
VALUES ('9780486264671', 4, 1);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780486264671', 5, 2);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780007113804', 6, 1);
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780670813029', 6, 1);
INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780486264671', 6, 1);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780670813029', 7, 2);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780553294385', 8, 1);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780345339706', 9, 4);

INSERT INTO is_part_of (isbn, order_code, number_of_copies)
VALUES ('9780486264671', 10, 2);
```

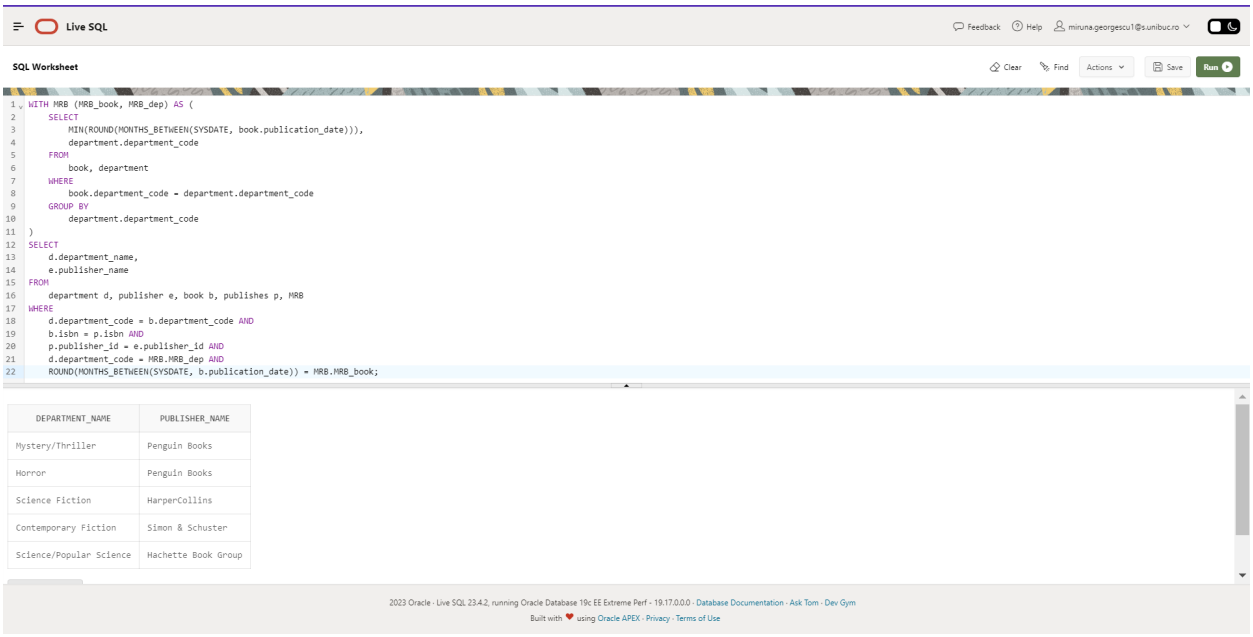
SQL Queries and Operations

Formulate and implement 5 complex SQL queries that will use, in their entirety, the following elements:

- *join operation on at least 4 tables*
- *row-level filtering*
- *synchronized subqueries involving at least 3 tables*
- *unsynchronized subqueries involving at least 3 tables*
- *data grouping, group functions, group-level filtering*
- *sorting*
- *use of at least 2 string functions, 2 calendar date functions, NVL and DECODE functions, and at least one CASE expression*
- *use of at least 1 query block (WITH clause)*

Recent Publications by Department

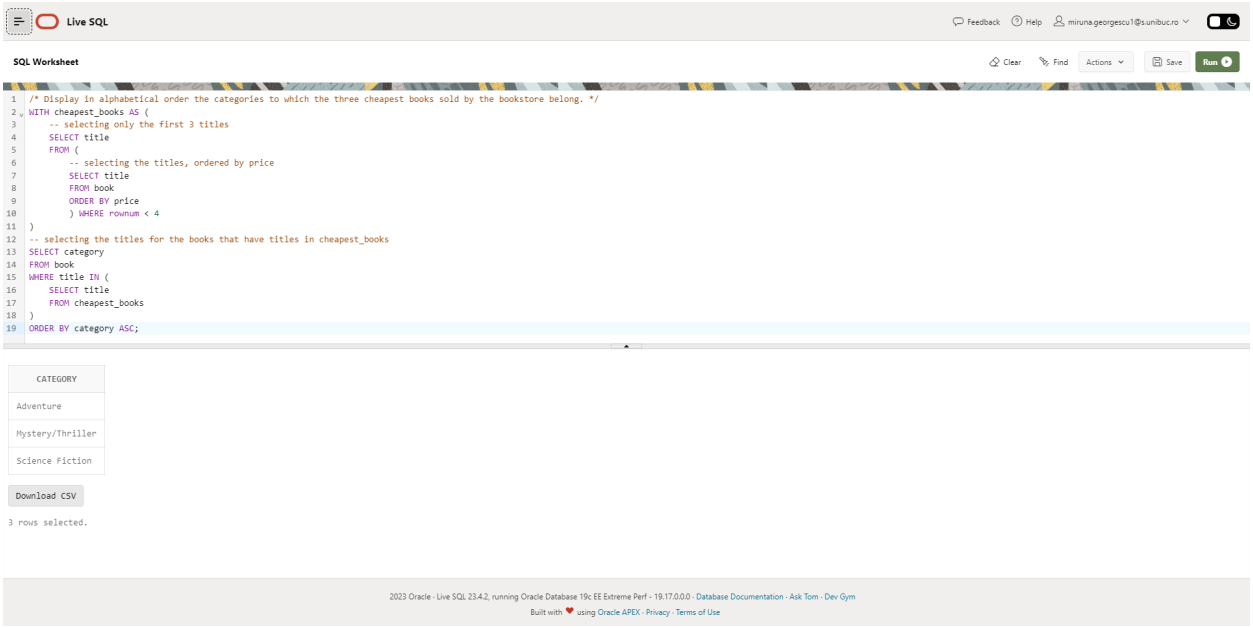
Display, for each department, the publisher where the most recent book was published.




```
-- MRB stands for Most Recent Book
WITH MRB (MRB_book, MRB_dep) AS (
    -- query block used to find the most recent publication date (the minimum number
    -- of months since publication) for each department
    SELECT
        MIN(ROUND(MONTHS_BETWEEN(SYSDATE, book.publication_date))),
        department.department_code
    FROM
        book, department
    WHERE
      -- joining the tables BOOK and DEPARTMENT on department_code
        book.department_code = department.department_code
    GROUP BY
        department.department_code
)
-- main query used to find the department name and the publishing house
-- name for the most recent books
SELECT
    d.department_name,
    e.publisher_name
FROM
    department d, publisher e, book b, publishes p, MRB
WHERE
    -- joining the tables based on department_code, isbn and publisher_id
    d.department_code = b.department_code AND
    b.isbn = p.isbn AND
    p.publisher_id = e.publisher_id AND
    -- filtering by department and by the number of months passed since publication
    d.department_code = MRB.MRB_dep AND
    ROUND(MONTHS_BETWEEN(SYSDATE, b.publication_date)) = MRB.MRB_book;
```

Categories for the Cheapest Books

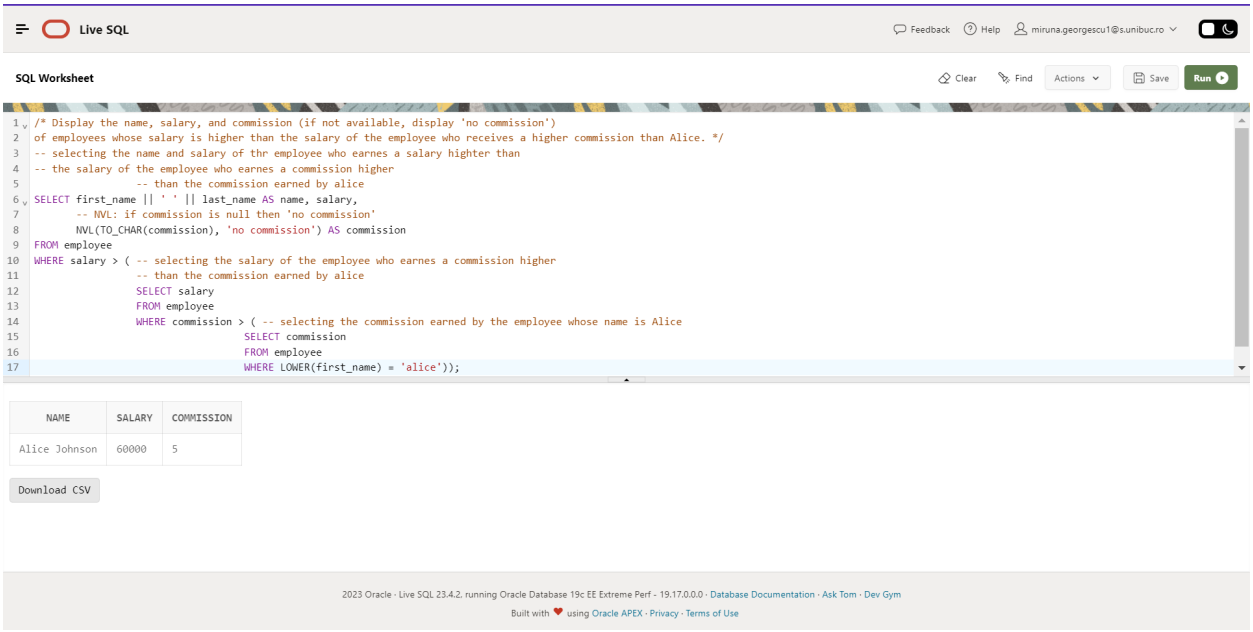
Display in alphabetical order the categories to which the three cheapest books sold by the bookstore belong.




```
WITH cheapest_books AS (
  -- selecting only the first 3 titles
    SELECT title
  FROM (
    -- selecting the titles, ordered by price
    SELECT title
    FROM book
    ORDER BY price
    ) WHERE rownum < 4
)
-- selecting the titles for the books that have titles in cheapest_books
SELECT category
FROM book
WHERE title IN (
  SELECT title
  FROM cheapest_books
)
ORDER BY category ASC;
```

Top Salaries

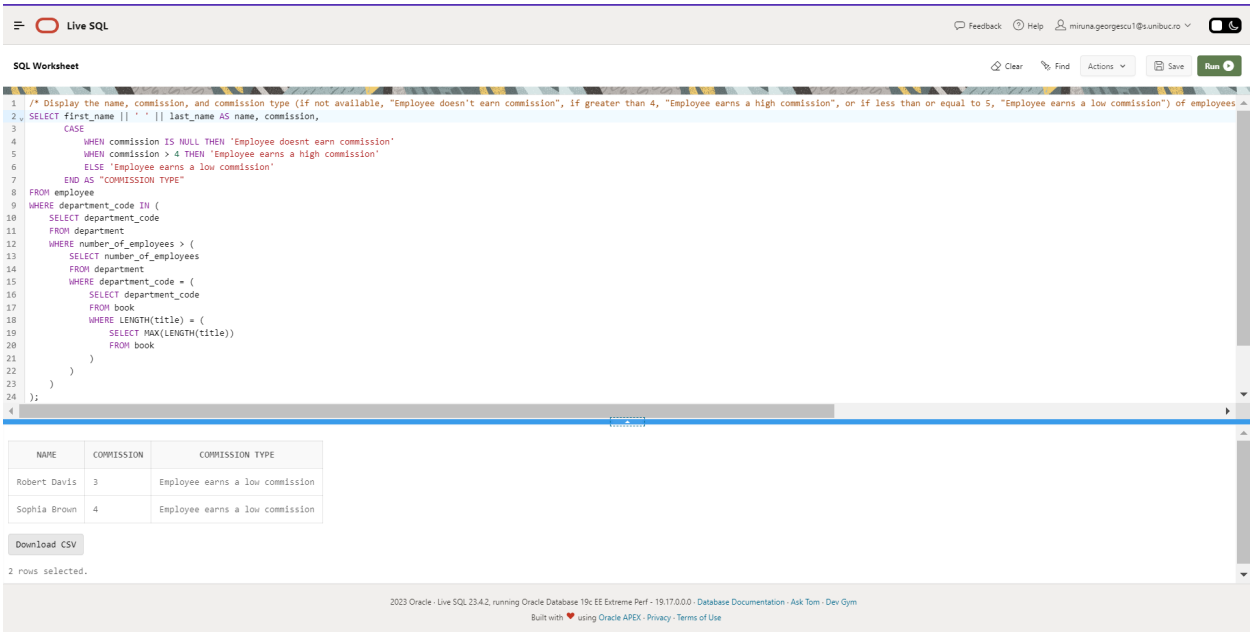
Display the name, salary, and commission (if not available, display 'no commission') of employees whose salary is higher than the salary of the employee who receives a higher commission than Alice.




```
-- selecting the name and salary of thr employee who earnes a salary highter than
-- the salary of the employee who earnes a commission higher
                -- than the commission earned by alice
SELECT first_name || ' ' || last_name AS name, salary,
       -- NVL: if commission is null then 'no commission'
       NVL(TO_CHAR(commission), 'no commission') AS commission
FROM employee
WHERE salary > ( -- selecting the salary of the employee who earnes a commission higher
                -- than the commission earned by alice
                SELECT salary
                FROM employee
                WHERE commission > ( -- selecting the commission earned by the employee
                                     -- whose name is Alice
                                    SELECT commission
                                    FROM employee
                                    WHERE LOWER(first_name) = 'alice'));
```

Commission Types

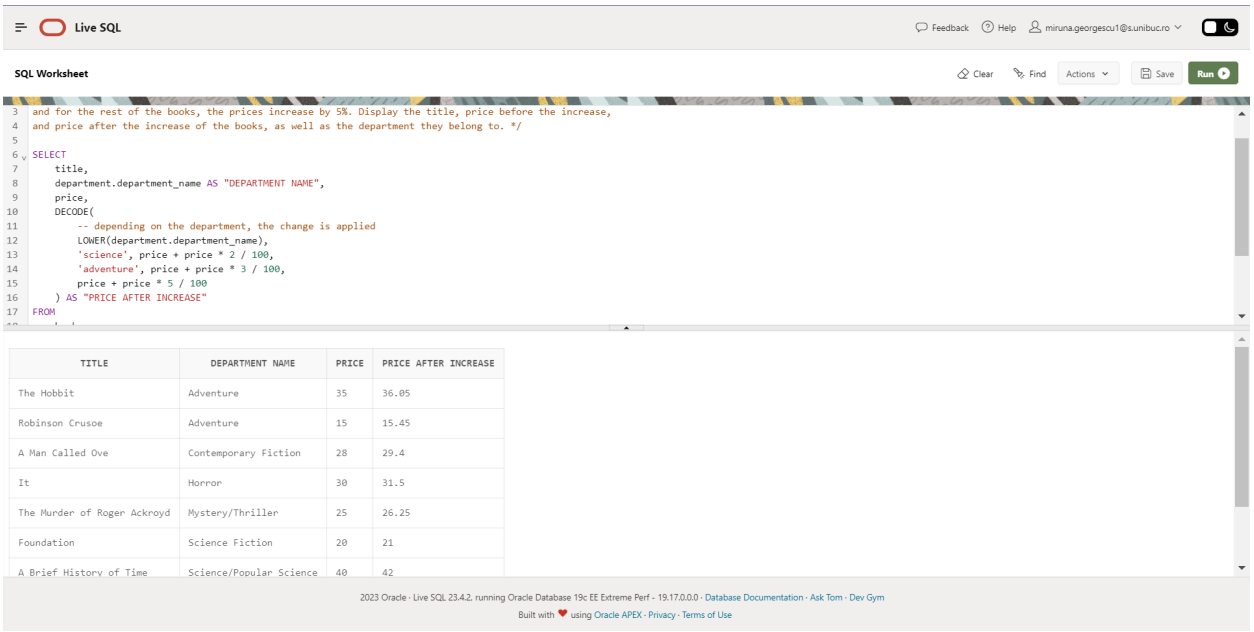
Display the name, commission, and commission type (if not available, "Employee doesn't earn commission", if greater than 4, "Employee earns a high commission", or if less than or equal to 5, "Employee earns a low commission") of employees working in departments with more employees than the department where the book with the longest title is located.




```
-- selecting the name and commission types for the employees
SELECT first_name || ' ' || last_name AS name, commission,
       CASE
           WHEN commission IS NULL THEN 'Employee doesnt earn commission'
           WHEN commission > 4 THEN 'Employee earns a high commission'
           ELSE 'Employee earns a low commission'
       END AS "COMMISSION TYPE"
FROM employee
WHERE department_code IN (
    -- selecting the department codes for the departments with more employees than the one selected prior
    SELECT department_code
    FROM department
    WHERE number_of_employees > (
      -- selecting the number of employees for the department
        SELECT number_of_employees
        FROM department
        WHERE department_code = (
        -- selecting the department code for the department in which the book with the longest title is in
            SELECT department_code
            FROM book
            WHERE LENGTH(title) = (
          -- selecting the length of the longest title
                SELECT MAX(LENGTH(title))
                FROM book
            )
        )
    )
);
```

Price Changes

Inflation causes the increase of all prices, and books are no exception. Therefore, the prices of books in the Science department increase by 2%, the prices of books in the Adventure department increase by 3%, and for the rest of the books, the prices increase by 5%. Display the title, price before the increase, and price after the increase of the books, as well as the department they belong to.



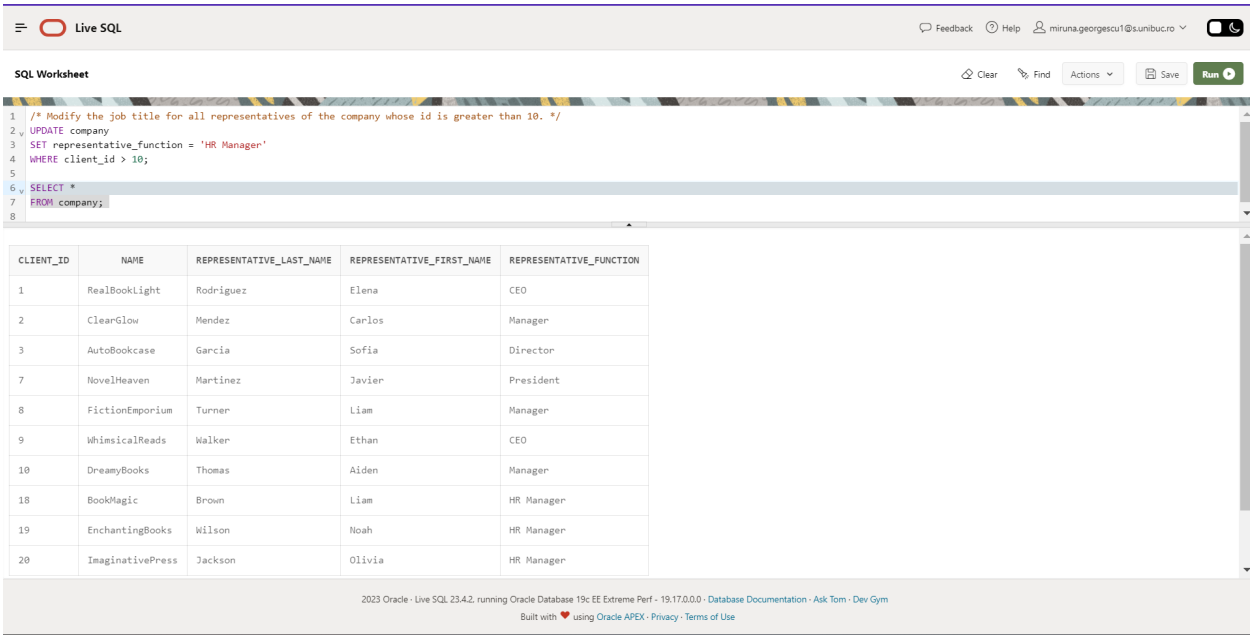

```
SELECT
    title,
    department.department_name AS "DEPARTMENT NAME",
    price,
    DECODE(
      -- depending on the department, the change is applied
        LOWER(department.department_name),
        'science', price + price * 2 / 100,
        'adventure', price + price * 3 / 100,
        price + price * 5 / 100
    ) AS "PRICE AFTER INCREASE"
FROM
    book
INNER JOIN
    -- joining the BOOK table and the DEPARTMENT table on department_code
    department ON department.department_code = book.department_code
ORDER BY
    -- ordering alphabetically by department name
    department.department_name ASC;
```

Operations Using Subqueries

Implementation of 3 data update and delete operations using subqueries.

Title Change

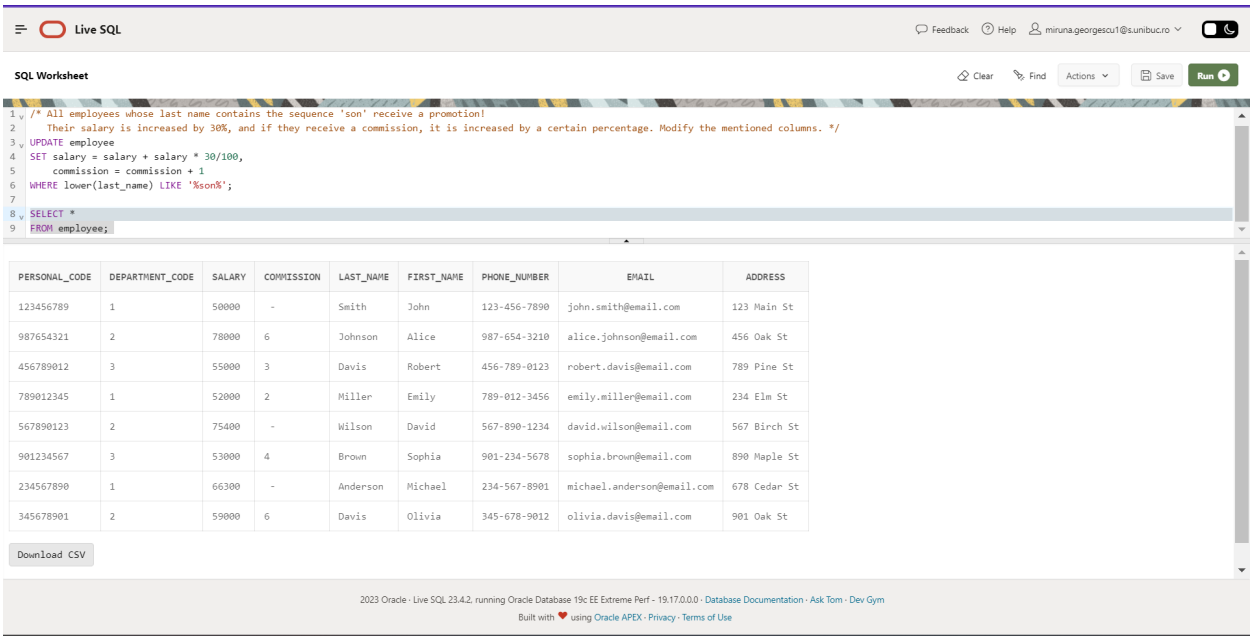
Modify the job title for all representatives of the company whose id is greater than 10.




```
UPDATE company
SET representative_function = 'HR Manager'
WHERE client_id > 10;
```

Employee Promotion

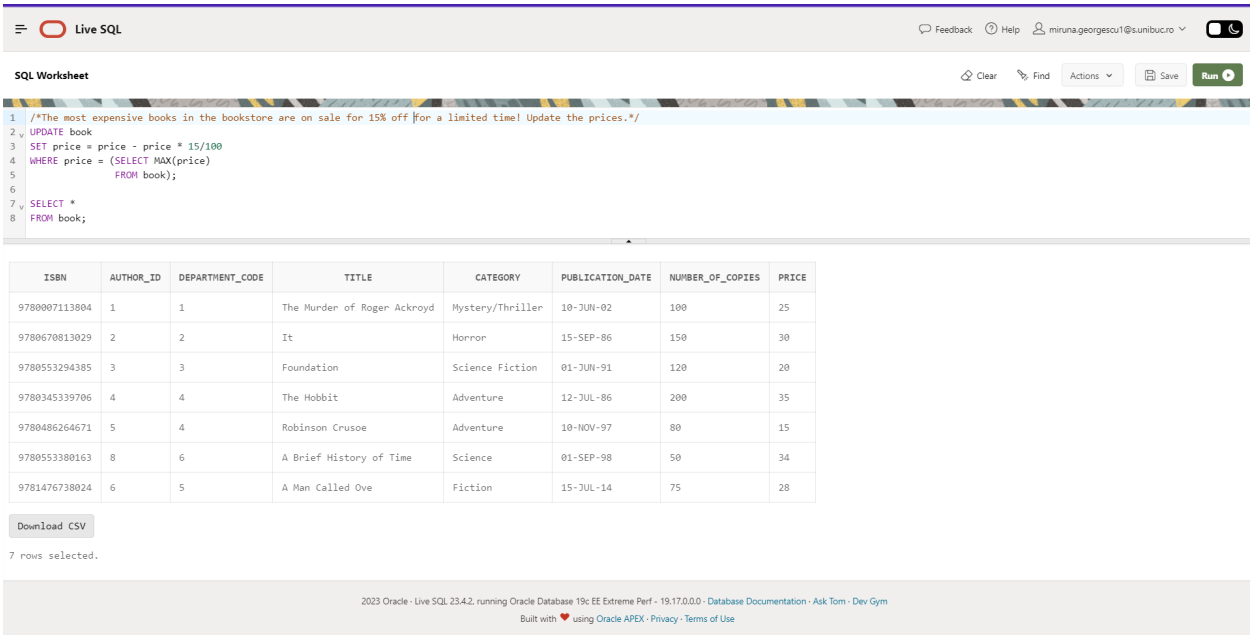
All employees whose last name contains the sequence 'son' receive a promotion!
Their salary is increased by 30%, and if they receive a commission, it is increased by a certain percentage. Modify the mentioned columns.




```
UPDATE employee
SET salary = salary + salary * 30/100,
    commission = commission + 1
WHERE lower(last_name) LIKE '%son%';
```

Sale

The most expensive books in the bookstore are on sale for 15% off for a limited time! Update the prices.



```
UPDATE book
SET price = price - price * 15/100
WHERE price = (SELECT MAX(price)
               FROM book);
```