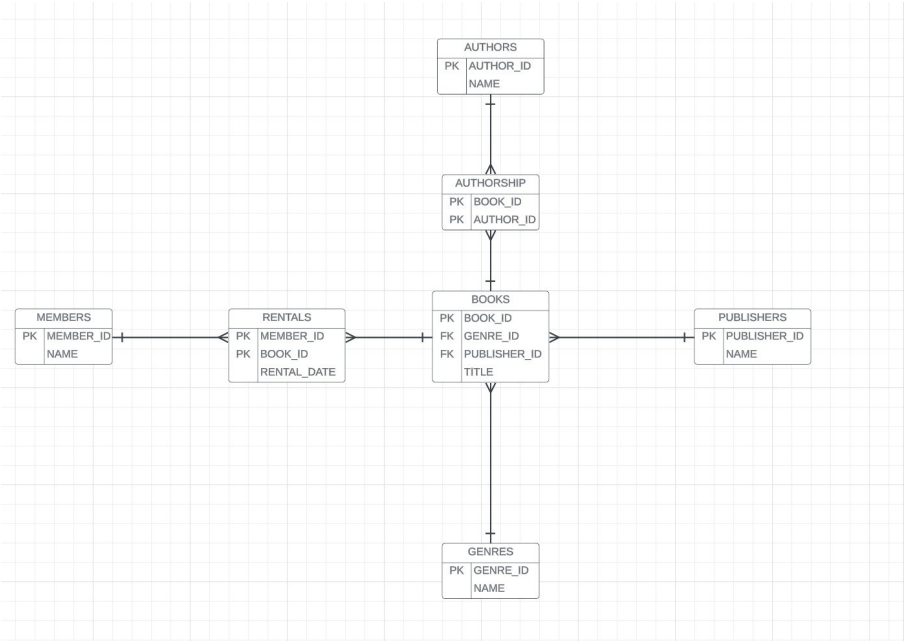
SISTEM GESTIONARE A DATELOR DESPRE IMPRUMUTURI DE CARTI

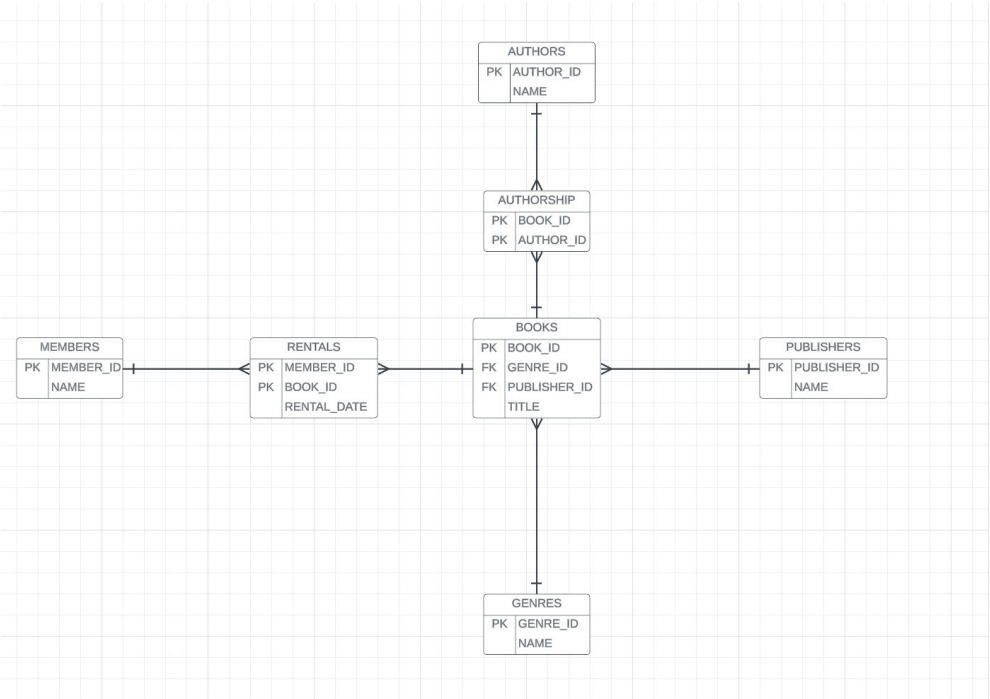
Nume: Georgescu Cosmin Gabriel
Grupa: 244

1.
Sistem de stocare a datelor despre imprumuturile de carti pe care le efectueaza membrii unei organizatii si informatii despre cartile respective.

2.



3.



4.
create table authors (author_id int primary key, name varchar2(255));

```
Table AUTHORS created.
```

create table publishers (publisher_id int primary key, name varchar2(255));

```
Table PUBLISHERS created.
```

create table genres (genre_id int primary key, name varchar2(255));

```
Table GENRES created.
```

create table members (member_id int primary key, name varchar2(255));

```
Table MEMBERS created.
```

create table books (book_id int primary key, genre_id int, publisher_id int, title varchar2(255), foreign key (genre_id) references genres (genre_id), foreign key (publisher_id) references publishers (publisher_id));

```
Table BOOKS created.
```

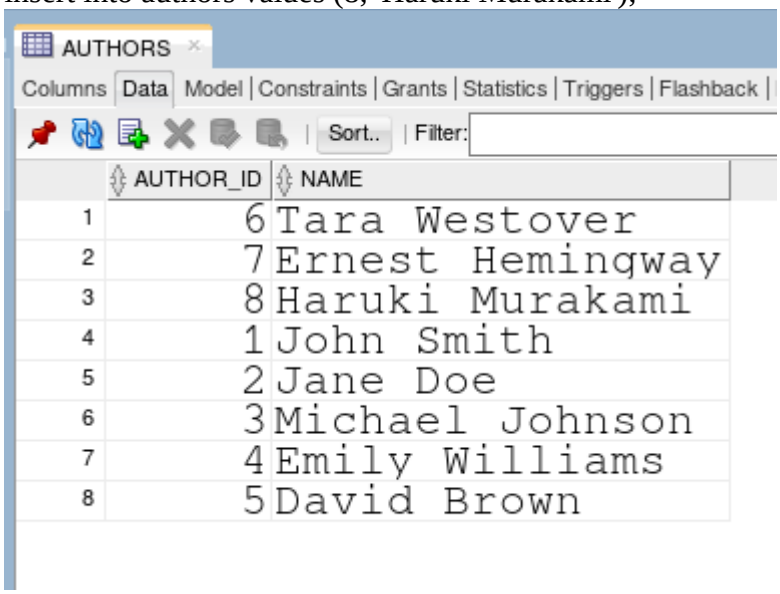
create table authorship (book_id int, author_id int, primary key (book_id, author_id), foreign key (book_id) references books (book_id), foreign key (author_id) references authors (author_id));

```
Table AUTHORSHIP created.
```

create table rentals (member_id int, book_id int, primary key (member_id, book_id), foreign key (member_id) references members (member_id), foreign key (book_id) references books (book_id), rental_date date);

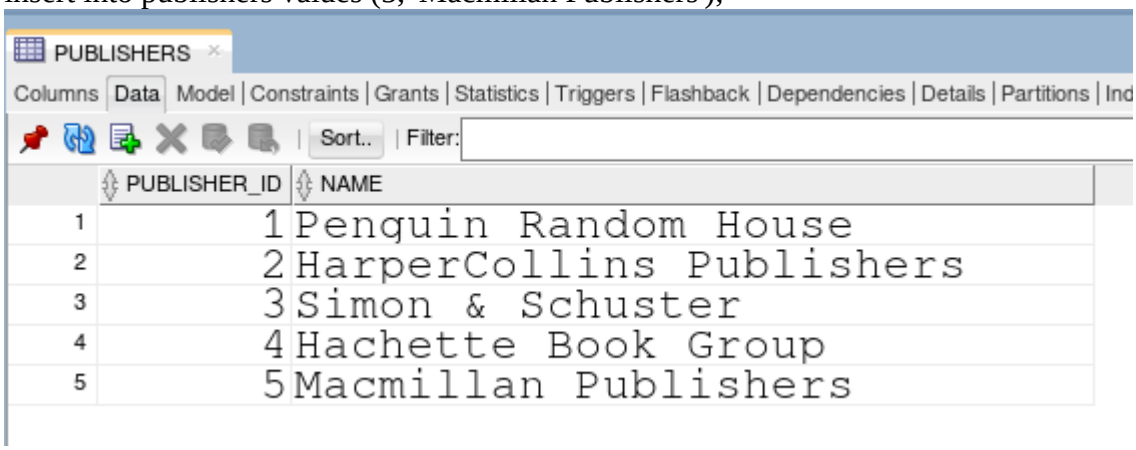
```
Table RENTALS created.
```

5.
insert into authors values (1, 'John Smith');
insert into authors values (2, 'Jane Doe');
insert into authors values (3, 'Michael Johnson');
insert into authors values (4, 'Emily Williams');
insert into authors values (5, 'David Brown');
insert into authors values (6, 'Tara Westover');
insert into authors values (7, 'Ernest Hemingway');
insert into authors values (8, 'Haruki Murakami');



	AUTHOR_ID	NAME
1	6	Tara Westover
2	7	Ernest Hemingway
3	8	Haruki Murakami
4	1	John Smith
5	2	Jane Doe
6	3	Michael Johnson
7	4	Emily Williams
8	5	David Brown

insert into publishers values (1, 'Penguin Random House');
insert into publishers values (2, 'HarperCollins Publishers');
insert into publishers values (3, 'Simon & Schuster');
insert into publishers values (4, 'Hachette Book Group');
insert into publishers values (5, 'Macmillan Publishers');



	PUBLISHER_ID	NAME
1	1	Penguin Random House
2	2	HarperCollins Publishers
3	3	Simon & Schuster
4	4	Hachette Book Group
5	5	Macmillan Publishers

insert into genres values (1, 'Fiction');
insert into genres values (2, 'Mystery');
insert into genres values (3, 'Romance');
insert into genres values (4, 'Science Fiction');
insert into genres values (5, 'Thriller');

GENRES

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashbac

Sort.. | Filter:

	GENRE_ID	NAME
1	1	Fiction
2	2	Mystery
3	3	Romance
4	4	Science Fiction
5	5	Thriller

insert into members values (1, 'Benjamin Thompson');
insert into members values (2, 'Maya Patel');
insert into members values (3, 'Lucas Anderson');
insert into members values (4, 'Olivia Ramirez');
insert into members values (5, 'Ethan Johnson');
insert into members values (6, 'Sophia Johnson');

MEMBERS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback |

Sort.. | Filter:

	MEMBER_ID	NAME
1	6	Sophia Johnson
2	1	Benjamin Thompson
3	2	Maya Patel
4	3	Lucas Anderson
5	4	Olivia Ramirez
6	5	Ethan Johnson

insert into books values (1, 1, 1, 'To Kill a Mockingbird');
insert into books values (2, 2, 2, '1984');
insert into books values (3, 3, 3, 'Pride and Prejudice');
insert into books values (4, 4, 4, 'The Great Gatsby');
insert into books values (5, 5, 5, 'The Catcher in the Rye');
insert into books values (6, 3, 2, 'Madame Bovary');
insert into books values (7, 1, 1, 'The Da Vinci Code');
insert into books values (8, 2, 2, 'Gone Girl');
insert into books values (9, 4, 3, 'The Hunger Games');
insert into books values (10, 3, 5, 'The Old Man and the Sea');
insert into books values (11, 1, 3, 'Kafka on the Shore');

BOOKS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	BOOK_ID	GENRE_ID	PUBLISHER_ID	TITLE
1	7	1	1	The Da Vinci Code
2	8	2	2	Gone Girl
3	9	4	3	The Hunger Games
4	10	3	5	The Old Man and the Sea
5	11	1	3	Kafka on the Shore
6	6	3	2	Madame Bovary
7	1	1	1	To Kill a Mockingbird
8	2	2	2	1984
9	3	3	3	Pride and Prejudice
10	4	4	4	The Great Gatsby
11	5	5	5	The Catcher in the Rye

insert into authorship values (1, 3);
insert into authorship values (1, 5);
insert into authorship values (2, 1);
insert into authorship values (2, 4);
insert into authorship values (3, 2);
insert into authorship values (4, 1);
insert into authorship values (4, 3);
insert into authorship values (5, 2);
insert into authorship values (5, 4);
insert into authorship values (5, 5);
insert into authorship values (5, 5);
insert into authorship values (7, 5);
insert into authorship values (8, 5);
insert into authorship values (9, 5);
insert into authorship values (10, 7);
insert into authorship values (11, 8);

AUTHORSHIP

Columns | Data | Model | Constraints | Grants | Stat

Sort.. | Filter:

	BOOK_ID	AUTHOR_ID
1	7	5
2	8	5
3	9	5
4	10	7
5	11	8
6	1	3
7	1	5
8	2	1
9	2	4
10	3	2
11	4	1
12	4	3
13	5	2
14	5	4
15	5	5

```
insert into rentals values (1, 3, to_date('2023-05-01', 'YYYY-MM-DD'));
insert into rentals values (1, 5, to_date('2023-05-02', 'YYYY-MM-DD'));
insert into rentals values (2, 1, to_date('2023-05-03', 'YYYY-MM-DD'));
insert into rentals values (2, 4, to_date('2023-05-04', 'YYYY-MM-DD'));
insert into rentals values (3, 2, to_date('2023-05-05', 'YYYY-MM-DD'));
insert into rentals values (3, 3, to_date('2023-05-06', 'YYYY-MM-DD'));
insert into rentals values (4, 1, to_date('2023-05-07', 'YYYY-MM-DD'));
insert into rentals values (4, 2, to_date('2023-05-08', 'YYYY-MM-DD'));
insert into rentals values (4, 5, to_date('2023-05-09', 'YYYY-MM-DD'));
insert into rentals values (5, 1, to_date('2023-05-10', 'YYYY-MM-DD'));
insert into rentals values (3, 6, to_date('2023-05-11', 'YYYY-MM-DD'));
insert into rentals values (5, 2, to_date('2023-05-12', 'YYYY-MM-DD'));
insert into rentals values (5, 3, to_date('2023-05-13', 'YYYY-MM-DD'));
insert into rentals values (6, 7, to_date('2023-05-18', 'YYYY-MM-DD'));
insert into rentals values (6, 8, to_date('2023-05-20', 'YYYY-MM-DD'));
insert into rentals values (6, 9, to_date('2023-05-21', 'YYYY-MM-DD'));
insert into rentals values (2, 11, to_date('2023-05-22', 'YYYY-MM-DD'));
```

RENTALS

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Fla

Sort.. | Filter:

	MEMBER_ID	BOOK_ID	RENTAL_DATE
1	2	11	22-MAY-23
2	6	9	21-MAY-23
3	6	8	20-MAY-23
4	6	7	18-MAY-23
5	5	3	13-MAY-23
6	5	2	12-MAY-23
7	3	6	11-MAY-23
8	5	1	10-MAY-23
9	4	5	09-MAY-23
10	4	2	08-MAY-23
11	4	1	07-MAY-23
12	3	3	06-MAY-23
13	3	2	05-MAY-23
14	2	4	04-MAY-23
15	2	1	03-MAY-23
16	1	5	02-MAY-23
17	1	3	01-MAY-23

6.
Preluati casele de publicatie si cartile pe care le-au publicat acestea.

```
create or replace procedure publishers_and_books is
   type publishers_t is table of publishers.publisher_id%type index by pls_integer;
   type publisher_books_t is table of books.title%type;
   publishers_arr publishers_t;
   publisher_books publisher_books_t := publisher_books_t();
   publisher_name publishers.name%type;
begin
   select publisher_id bulk collect into publishers_arr from publishers;
   for i in publishers_arr.first..publishers_arr.last loop
      select name into publisher_name from publishers where publisher_id = publishers_arr(i);
      dbms_output.put_line(publisher_name);
      select title bulk collect into publisher_books from books where publisher_id =
publishers_arr(i);
      for j in publisher_books.first..publisher_books.last loop
         dbms_output.put_line(' - ' || publisher_books(j));
      end loop;
   end loop;
end;
```

```
/
execute publishers_and_books();
/
```

Script Output x Query Result x

Task completed in 0.048 seconds

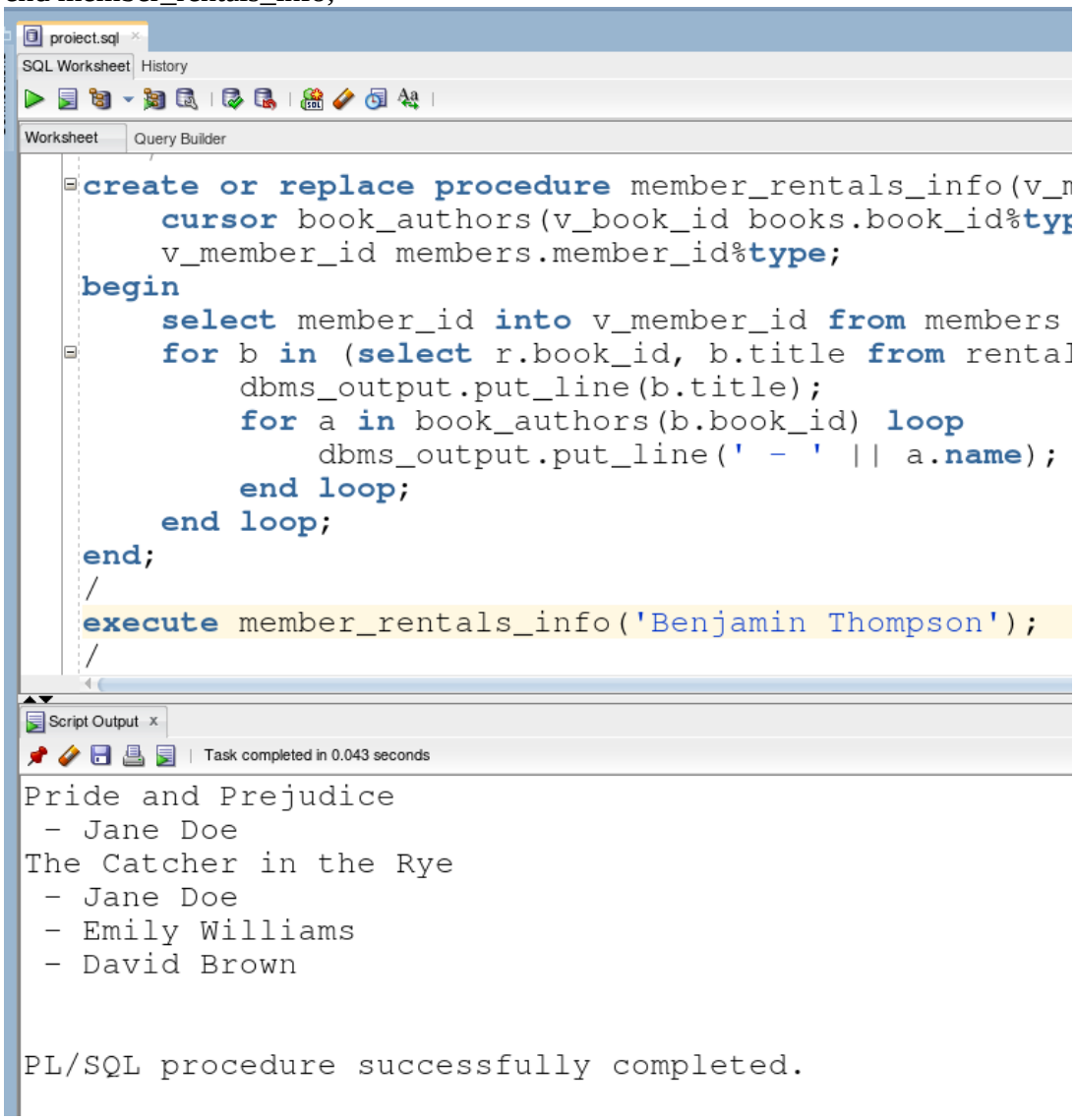
```
Penguin Random House
 - The Da Vinci Code
 - To Kill a Mockingbird
HarperCollins Publishers
 - Gone Girl
 - Madame Bovary
 - 1984
Simon & Schuster
 - The Hunger Games
 - Kafka on the Shore
 - Pride and Prejudice
Hachette Book Group
 - The Great Gatsby
Macmillan Publishers
 - The Old Man and the Sea
 - The Catcher in the Rye


PL/SQL procedure successfully completed.
```

7.
Preluați toate cărțile și lista completă a autorilor respectivi inchiriate de către un membru a cărui identitate este dată pe baza nume.

```
create or replace procedure member_rentals_info(v_member_name in members.name%type) is
   cursor book_authors(v_book_id books.book_id%type) is select a2.name from authorship a1 join
authors a2 on a1.author_id = a2.author_id where a1.book_id = v_book_id;
   v_member_id members.member_id%type;
begin
   select member_id into v_member_id from members where name = v_member_name;
   for b in (select r.book_id, b.title from rentals r join books b on r.book_id = b.book_id where
r.member_id = v_member_id) loop
      dbms_output.put_line(b.title);
      for a in book_authors(b.book_id) loop
         dbms_output.put_line(' - ' || a.name);
```

```
        end loop;
    end loop;
end member_rentals_info;
```



8.
Preluati genul preferat al unui membru identificat prin nume. Daca membrul este autor sau are mai multe genuri preferate, se va ridica o eroare.

```
create or replace function member_prefered_genre(v_member_name in members.name%type)
return genres.name%type is
    v_member_id members.member_id%type;
    v_max number;
    v_occurences number;
    return_t genres.name%type;
begin
    declare
        flag number := 0;
    begin
        SELECT count(*) into flag
        FROM authors
        WHERE name = v_member_name;
```

```
        if flag != 0 then raise_application_error(-20001, 'can not spy on authors');
        end if;
    end;
    select member_id into v_member_id from members where name = v_member_name;
    select genre, occurence into return_t, v_max from (select g.name as genre, count(g.name) as occurence from rentals r join books b on r.book_id = b.book_id join genres g on b.genre_id = g.genre_id where r.member_id = v_member_id group by g.name order by occurence desc) where rownum <= 1;
    select count(*) into v_occurences from (select count(g.name) as occurence from rentals r join books b on r.book_id = b.book_id join genres g on b.genre_id = g.genre_id where r.member_id = v_member_id group by g.name order by occurence desc) where occurence = v_max;
    if v_occurences > 1 then raise_application_error(-20002, 'more than one prefered genre');
    end if;
    return return_t;
end member_prefered_genre;
```

```
begin
    dbms_output.put_line(member_prefered_genre('Lucas Anderson'));
end;
```

Script Output | Query Result

Task completed in 0.053 seconds

```
Romance


PL/SQL procedure successfully completed.
```

```
/
begin
    dbms_output.put_line(member_prefered_genre('Jane Doe'));
end;
```

Script Output | Query Result

Task completed in 0.083 seconds

```
Error starting at line : 99 in command -
begin
    dbms_output.put_line(member_prefered_genre('Jane Doe'));
end;
Error report -
ORA-20001: can not spy on authors
ORA-06512: at "GRUPA244.MEMBER_PREFERED_GENRE", line 13
ORA-06512: at line 2


Function MEMBER_PREFERED_GENRE compiled
```

```
begin
    dbms_output.put_line(member_prefered_genre('Ethan Johnson'));
end;
```

Script Output | Query Result

Task completed in 0.089 seconds

```
Error starting at line : 102 in command -
begin
    dbms_output.put_line(member_prefered_genre('Ethan Johnson'));
end;
Error report -
ORA-20002: more than one prefered genre
ORA-06512: at "GRUPA244.MEMBER_PREFERED_GENRE", line 19
ORA-06512: at line 2
```

9.
Preluati fanul numarul 1 pentru fiecare autor.

```
create or replace procedure authors_and_fan is
   type authors_t is table of authors.name%type index by pls_integer;
   col authors_t;
   buba authors.name%type;
begin
   for p in (SELECT a.author_id, a.name AS author, f.name AS fan
   FROM authors a
   JOIN (
      SELECT au.author_id, m.member_id AS fan_id, m.name, dense_rank() OVER (PARTITION
BY au.author_id ORDER BY COUNT(*) DESC) AS rn
      FROM authorship au
      JOIN books b ON au.book_id = b.book_id
      JOIN rentals r ON b.book_id = r.book_id
      JOIN members m ON r.member_id = m.member_id
      GROUP BY au.author_id, m.member_id, m.name
   ) f ON a.author_id = f.author_id AND f.rn = 1)
    loop
      if col.exists(p.author_id) then
        raise TOO_MANY_ROWS;
      else
         col(p.author_id) := p.fan;
      end if;
      dbms_output.put_line(p.author || ' -> ' || p.fan);
   end loop;
   for i in col.first..col.last loop
      buba := col(i);
   end loop;
   exception
      when NO_DATA_FOUND then dbms_output.put_line('An author has no fans :(');
      when TOO_MANY_ROWS then dbms_output.put_line('An author has more than one biggest
fan');
end;
```

```
end;
/
execute authors_and_fan();
/
```

Script Output x Query Result x

Task completed in 0.049 seconds

```
John Smith -> Maya Patel
An author has more than one biggest fan


PL/SQL procedure successfully completed.
```

```
    ) f ON a.author_id = f.author_id AND f.rn = 1)
     loop
        --if col.exists(p.author_id) then
          --raise TOO_MANY_ROWS;
        --else
            col(p.author_id) := p.fan;
        --end if;
        dbms_output.put_line(p.author || ' -> ' || p.fan);
```

Script Output x Query Result x

Task completed in 0.05 seconds

```
Procedure AUTHORS_AND_FAN compiled

John Smith -> Maya Patel
John Smith -> Lucas Anderson
John Smith -> Olivia Ramirez
John Smith -> Ethan Johnson
Jane Doe -> Benjamin Thompson
Michael Johnson -> Maya Patel
Emily Williams -> Olivia Ramirez
David Brown -> Sophia Johnson
Haruki Murakami -> Maya Patel
An author has no fans :(


PL/SQL procedure successfully completed.
```

10.

```
CREATE OR REPLACE TRIGGER sus_activity BEFORE INSERT ON rentals
declare
  current_hour NUMBER;
begin
  select EXTRACT(hour from SYSTIMESTAMP) into current_hour from DUAL;
  if current_hour >= 15 or current_hour < 8 then RAISE_APPLICATION_ERROR(-20001, 'Cannot
insert into rentals between 6PM and 8AM');
```

```
  end if;
end;
```

```
insert into rentals values (2, 11, to_date('2023-05-22', 'YYYY-MM-
INSERT INTO rentals VALUES (1, 6, SYSTIMESTAMP);
/
```

Script Output | Query Result

Task completed in 0.079 seconds

```
Error starting at line : 77 in command -
INSERT INTO rentals VALUES (1, 6, SYSTIMESTAMP)
Error report -
ORA-20001: Cannot insert into rentals between 6PM and 8AM
ORA-06512: at "GRUPA244.SUS_ACTIVITY", line 5
ORA-04088: error during execution of trigger 'GRUPA244.SUS_ACTIVITY'
```

11.

```
CREATE OR REPLACE TRIGGER check_rental_date BEFORE UPDATE ON rentals FOR EACH ROW
BEGIN
  IF (:NEW.rental_date <= :OLD.rental_date) THEN RAISE_APPLICATION_ERROR(-20001, 'Updated rental date cannot be before the old date.');
  END IF;
END;
```

```
/
update rentals set rental_date = to_date('2023-04-22', 'YYYY-MM-DD') where member_id = 2 and book_id = 11;
/
```

Script Output | Query Result

Task completed in 0.077 seconds

```
Error starting at line : 167 in command -
update rentals set rental_date = to_date('2023-04-22', 'YYYY-MM-DD') where member_id = 2 and book_id = 11
Error report -
ORA-20001: Updated rental date cannot be before the old date.
ORA-06512: at "GRUPA244.CHECK_RENTAL_DATE", line 2
ORA-04088: error during execution of trigger 'GRUPA244.CHECK_RENTAL_DATE'
```

12.

```
create or replace TRIGGER unauthorized_acc before alter or drop on schema
begin
   if user != 'SYS' then RAISE_APPLICATION_ERROR(-20001, 'Unauthorized drop or alter query from non-sys users');
   else dbms_output.put_line('Query authorized');
   end if;
end;
```

```
/
drop table authors;
```

Script Output x Query Result x

Task completed in 0.078 seconds

```
Error starting at line : 173 in command -
drop table authors
Error report -
ORA-00604: error occurred at recursive SQL level 1
ORA-20001: Unauthorized drop or alter query from non-sys users
ORA-06512: at line 2
00604. 00000 -  "error occurred at recursive SQL level %s"
*Cause:    An error occurred while processing a recursive SQL statement
           (a statement applying to internal dictionary tables).
*Action:   If the situation described in the next error on the stack
           can be corrected, do so; otherwise contact Oracle Support.
```