BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

CYBERNETICS, STATISTICS AND ECONOMIC INFORMATICS FACULTY

DBMS PROJECT

Managing a Cat Contest

Coordinator:

Prof. Univ. Dr. Diaconiţa V Vlad

Student: COSTAN CRISTIANA

Group: 1066

Bucharest

2023

CONTENTS

DESCRIPTION OF THE PROJECT

My project was created having in mind the idea of managing a cat contest. Due to the large number of data about the cats, judges and owners, one can manage more easily with a database the ranking and grouping of the cats based on their performance and characteristics along with the sorting of the judges based on their years of experience or other attributes related to their profession.

The relationships between tables can be noticed in the database schema shown in the following chapter. When it comes to speaking about owners, only one owner can come from each registered address and an address can belong to one owner only, being an one to one relationship between the tables addresses and owners. This type of relationship was chosen as a precautionary measure: we long for diversity and we do not seek to cause hatred between neighbors.

Speaking of relationships, an one to many relationship is the one between the cats table and the owners table: a cat can have an owner, while an owner can have many cats. Another one to many relationship is the recursive relationship from the judges table, based on the manager ID, which also establishes a hierarchy between the judges. The rest of the relationships are one to one: a cat can have only one place in the contest and a place in the contest can be occupied by only one cat. Moreover, a cat can be judged only by one judge and a judge can arbitrate only one cat, this being a solid refference to traditional judging practiced in Europe.

Due to having a small number of cats, the contest will focus mainly on evaluating them together and we will not care about their breeds or sizes when it comes to their final grades. We will only take into account their distinctive characteristics (breed, sex, size, fur type) when we either need to group them separately before the contest to avoid possible accidents or to study the participations rate of one particularity compared to another.

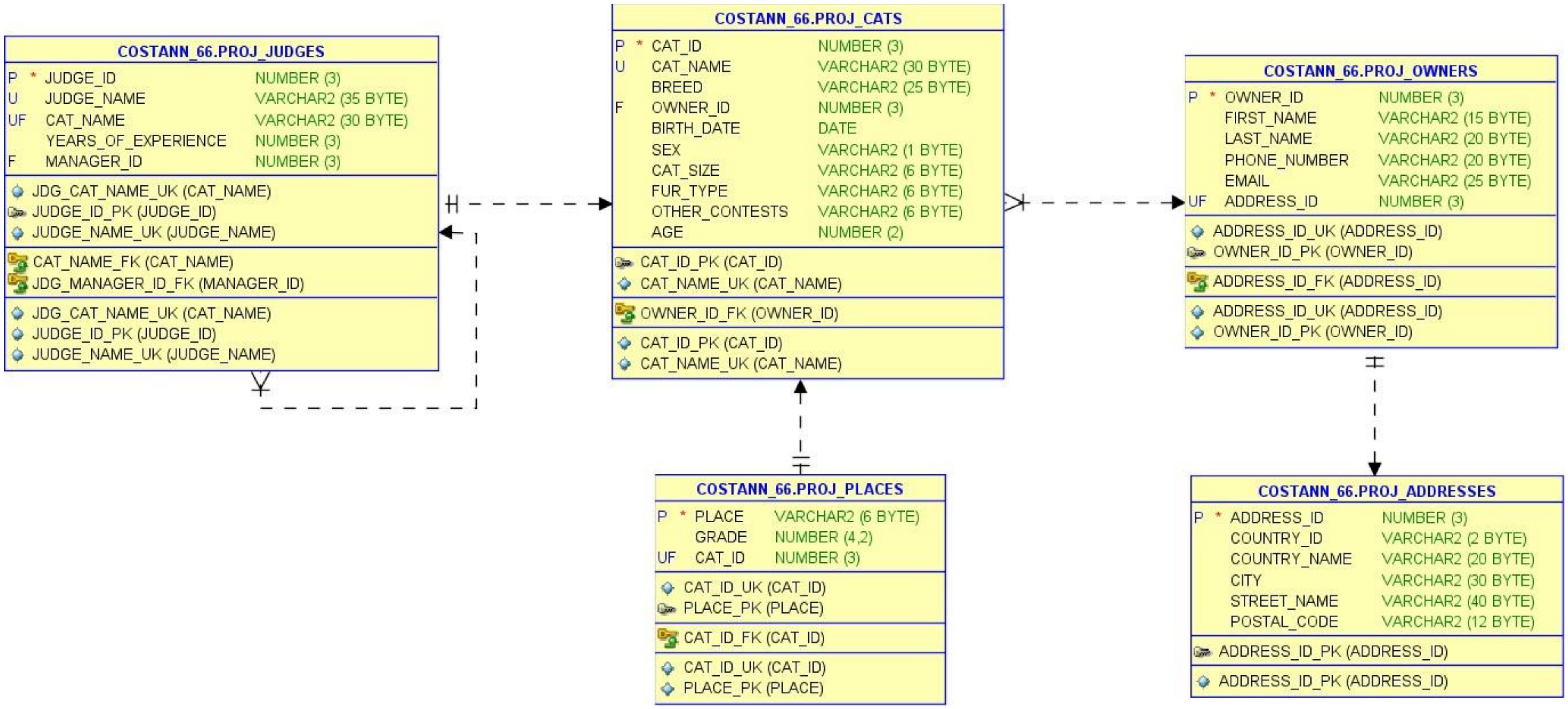
As in every cat contest, the cats have scene names, therefore they need to be unique. Furthermore, the judges that participate in the contest need to judge only one cat, hence their name uniqueness. Because I have taken into consideration the possibility of having one or more owners show up with two or more cats because of their kinship, the owner ID coresponding to the registered cat is not unique.

Additionally, ID related data from the tables judges (e.g. judge name), cats (e.g. cat name) and addresses (e.g. country ID) can be classified independently from other tables by data from the same table, while the others require data from other tables for their grouping to make sense.

Of course, we cannot forget the awards that a contest is well-known for. That being said, this competition has many title awards and trophies for the cats, given the fact that participants do not join for money.

Although some required statements fit better with the use of different numbers, I have tried my best to use them on my data that is mainly composed of cats’ specific features or owners’ and judges’ information.

THE DATABASE SCHEMA



COSTANN_66.PROJ_JUDGES
P * JUDGE_ID NUMBER (3)
U JUDGE_NAME VARCHAR2 (35 BYTE)
UF CAT_NAME VARCHAR2 (30 BYTE)
YEARS_OF_EXPERIENCE NUMBER (3)
F MANAGER_ID NUMBER (3)
JDG_CAT_NAME_UK (CAT_NAME)
JUDGE_ID_PK (JUDGE_ID)
JUDGE_NAME_UK (JUDGE_NAME)
CAT_NAME_FK (CAT_NAME)
JDG_MANAGER_ID_FK (MANAGER_ID)
JDG_CAT_NAME_UK (CAT_NAME)
JUDGE_ID_PK (JUDGE_ID)
JUDGE_NAME_UK (JUDGE_NAME)
COSTANN_66.PROJ_CATS
P * CAT_ID NUMBER (3)
U CAT_NAME VARCHAR2 (30 BYTE)
BREED VARCHAR2 (25 BYTE)
F OWNER_ID NUMBER (3)
BIRTH_DATE DATE
SEX VARCHAR2 (1 BYTE)
CAT_SIZE VARCHAR2 (6 BYTE)
FUR_TYPE VARCHAR2 (6 BYTE)
OTHER_CONTESTS VARCHAR2 (6 BYTE)
AGE NUMBER (2)
CAT_ID_PK (CAT_ID)
CAT_NAME_UK (CAT_NAME)
OWNER_ID_FK (OWNER_ID)
CAT_ID_PK (CAT_ID)
CAT_NAME_UK (CAT_NAME)
COSTANN_66.PROJ_OWNERS
P * OWNER_ID NUMBER (3)
FIRST_NAME VARCHAR2 (15 BYTE)
LAST_NAME VARCHAR2 (20 BYTE)
PHONE_NUMBER VARCHAR2 (20 BYTE)
EMAIL VARCHAR2 (25 BYTE)
UF ADDRESS_ID NUMBER (3)
ADDRESS_ID_UK (ADDRESS_ID)
OWNER_ID_PK (OWNER_ID)
ADDRESS_ID_FK (ADDRESS_ID)
ADDRESS_ID_UK (ADDRESS_ID)
OWNER_ID_PK (OWNER_ID)
COSTANN_66.PROJ_PLACES
P * PLACE VARCHAR2 (6 BYTE)
GRADE NUMBER (4,2)
UF CAT_ID NUMBER (3)
CAT_ID_UK (CAT_ID)
PLACE_PK (PLACE)
CAT_ID_FK (CAT_ID)
CAT_ID_UK (CAT_ID)
PLACE_PK (PLACE)
COSTANN_66.PROJ_ADDRESSES
P * ADDRESS_ID NUMBER (3)
COUNTRY_ID VARCHAR2 (2 BYTE)
COUNTRY_NAME VARCHAR2 (20 BYTE)
CITY VARCHAR2 (30 BYTE)
STREET_NAME VARCHAR2 (40 BYTE)
POSTAL_CODE VARCHAR2 (12 BYTE)
ADDRESS_ID_PK (ADDRESS_ID)
ADDRESS_ID_PK (ADDRESS_ID)

Figure 1. Schema from semester 1 (no modifications made in semester 2)

SOLVED PROBLEMS

1. **Modify a cat's grade using CASE. If a cat has a grade below 5, 2 points are added, if it is below 7, 1 point is added, otherwise leave the grade as it is. For exemplification, take the grade of cat number 4.**

Solution:

```
DECLARE
v_grade proj_places.grade%TYPE;
add_point NUMBER(1);
BEGIN
  SELECT grade INTO v_grade from proj_places where cat_id=4;
DBMS_OUTPUT.PUT_LINE('The initial grade is: '||v_grade);
add_point :=
  CASE
   WHEN v_grade < 5 THEN 2
   WHEN v_grade BETWEEN 5 AND 7 THEN 1
   ELSE 0
  END;
v_grade := v_grade + add_point;
DBMS_OUTPUT.PUT_LINE('The final grade is: '||v_grade);
END;
/
```

2. **Sort the judges by their experience using CASE. If they have 1 or under 1 year of experience, their category is "3rd Category"; if they have more than 1 year but less than 3 years, their category is "2nd Category"; if they have 3 or more than 3 years and less than 6 years, their category is "1st Category"; if they have 6 or more than 6 years they are "Master".**

Solution:

```
DECLARE
   v_category VARCHAR2(20);
   CURSOR cur_exp IS SELECT judge_name, years_of_experience FROM proj_judges ORDER
BY years_of_experience;
   v_exp_record cur_exp%ROWTYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE('The judges are: ');
OPEN cur_exp;
LOOP
FETCH cur_exp INTO v_exp_record;
EXIT WHEN cur_exp%NOTFOUND;
   CASE
```

```
    WHEN v_exp_record.years_of_experience <= 1 THEN v_category := '3rd Category';
    WHEN v_exp_record.years_of_experience > 1 AND v_exp_record.years_of_experience < 3
THEN v_category := '2nd Category';
    WHEN v_exp_record.years_of_experience >= 3 AND v_exp_record.years_of_experience < 6
THEN v_category := '1st Category';
    WHEN v_exp_record.years_of_experience >= 6 THEN v_category := 'Master';
    ELSE v_category := 'Undefined';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(v_exp_record.judge_name||' - '||v_category);
  END LOOP;
CLOSE cur_exp;
END;
/
```

3. **Display the participants according to their country of origin.**

Solution:

```
DECLARE
  CURSOR cur_country IS SELECT country_id, first_name, last_name, country_name FROM
proj_addresses a, proj_owners o WHERE a.address_id = o.address_id ORDER BY country_name;
  v_country_record cur_country%ROWTYPE;
BEGIN
OPEN cur_country;
LOOP
FETCH cur_country INTO v_country_record;
EXIT WHEN cur_country%NOTFOUND;
   IF v_country_record.country_id='RO' THEN
   DBMS_OUTPUT.PUT_LINE(v_country_record.first_name||' '||v_country_record.last_name||' is
from Romania.');
   ELSIF v_country_record.country_id='MD' THEN
   DBMS_OUTPUT.PUT_LINE(v_country_record.first_name||' '||v_country_record.last_name||' is
from Moldova.');
   ELSIF v_country_record.country_id='UK' THEN
   DBMS_OUTPUT.PUT_LINE(v_country_record.first_name||' '||v_country_record.last_name||' is
from United Kingdom.');
   ELSE
   DBMS_OUTPUT.PUT_LINE('We do not have a record of this country.');
  END IF;
END LOOP;
CLOSE cur_country;
END;
/
```

4. **Display the judges that have the id between 3 and 8 as long as they have more than 1 year of experience, which is the minimum possible number for the years of experience.**

Solution:

```
DECLARE
v_years_of_exp proj_judges.years_of_experience%TYPE;
v_exp v_years_of_exp%TYPE;
BEGIN
SELECT MIN(years_of_experience) INTO v_exp FROM proj_judges;
FOR i in 3..8 LOOP
SELECT years_of_experience INTO v_years_of_exp FROM proj_judges WHERE judge_id=i;
   IF v_years_of_exp > 1 THEN
     dbms_output.put_line('The judge with ID '||i||' has more than 1 year of experience.');
   END IF;
END LOOP;
END;
/
```

5. **Display the cats with odd IDs.**

Solution:

```
DECLARE
v_cat_name proj_cats.cat_name%TYPE;
i NUMBER := 1;
BEGIN
dbms_output.put_line('The cats with odd IDs are:');
WHILE i <= 13 LOOP
SELECT cat_name INTO v_cat_name FROM proj_cats WHERE cat_id=i;
dbms_output.put_line(i||'-->'||v_cat_name);
i:=i+2;
END LOOP;
END;
/
```

6. **Display the email of every owner using an index by table.**

Solution:

```
DECLARE
  TYPE t_email_rec IS TABLE OF proj_owners%ROWTYPE
  INDEX BY BINARY_INTEGER;
  v_email_rec_tab t_email_rec;
BEGIN
DBMS_OUTPUT.PUT_LINE('The email addresses of the owners are:');
FOR email_rec IN (SELECT * FROM proj_owners)
LOOP
```

```
  v_email_rec_tab(email_rec.owner_id) := email_rec;
  DBMS_OUTPUT.PUT_LINE(v_email_rec_tab(email_rec.owner_id).first_name||' -->
'||v_email_rec_tab(email_rec.owner_id).email);
END LOOP;
END;
/
```

7. **Display how many years of experience has every judge using an index by table.**

Solution:

```
DECLARE
  TYPE t_judge_rec IS TABLE OF proj_judges%ROWTYPE
  INDEX BY PLS_INTEGER;
  v_judge_rec_tab t_judge_rec;
BEGIN
FOR judge_rec IN (SELECT * FROM proj_judges ORDER BY years_of_experience)
LOOP
  v_judge_rec_tab(judge_rec.judge_id) := judge_rec;
  DBMS_OUTPUT.PUT_LINE(v_judge_rec_tab(judge_rec.judge_id).judge_name||' has been a
judge for '||v_judge_rec_tab(judge_rec.judge_id).years_of_experience||' years.');
END LOOP;
END;
/
```

8. **Display where each cat got awarded at the end of the contest using a nested table. If they got 1st, 2nd or 3rd place, mention they got crowned and awarded on the main stage. Otherwise, mention they got a diploma on the second stage.**

Solution:

```
DECLARE
  TYPE t_place_rec IS RECORD(
  cat_name proj_cats.cat_name%TYPE,
  place proj_places.place%TYPE);

  TYPE t_ibt IS TABLE OF t_place_rec;
  t t_ibt;
  i PLS_INTEGER;
BEGIN
 SELECT c.cat_name, p.place BULK COLLECT INTO t
   FROM proj_cats c, proj_places p
   WHERE c.cat_id = p.cat_id
   ORDER BY grade DESC;

  i:=t.FIRST;
  WHILE i IS NOT NULL LOOP
     IF t(i).place IN ('1st','2nd','3rd') THEN
     DBMS_OUTPUT.PUT_LINE(t(i).cat_name||' got crowned and awarded on the main stage.');
```

```
    ELSE
    DBMS_OUTPUT.PUT_LINE(t(i).cat_name||' got a diploma on the second stage.');
    END IF;
    i:=t.NEXT(i);
  END LOOP;
END;
/
```

9. **Display where each owner lives (full address) and how we can contact them (email and phone number) using a nested table.**

Solution:

```
DECLARE
  TYPE t_owners_rec IS RECORD(
  full_name proj_owners.last_name%TYPE,
  email proj_owners.email%TYPE,
  phone_number proj_owners.phone_number%TYPE,
  city proj_addresses.city%TYPE,
  country_name proj_addresses.country_name%TYPE);

  TYPE t_ibt IS TABLE OF t_owners_rec;
  t t_ibt;
  i PLS_INTEGER;
BEGIN
 SELECT first_name||' '||last_name AS full_name, email, phone_number, city, country_name
BULK COLLECT INTO t
   FROM proj_owners o, proj_addresses a
   WHERE o.address_id = a.address_id
   ORDER BY last_name;

   DBMS_OUTPUT.PUT_LINE('We have '||t.COUNT||' owners in our contest:');

  i:=t.FIRST;
  WHILE i IS NOT NULL LOOP
    DBMS_OUTPUT.PUT_LINE(t(i).full_name||' lives in '||t(i).city||', '||t(i).country_name||' and can
be found at '||t(i).email||' or '||t(i).phone_number||'.');
    i:=t.NEXT(i);
  END LOOP;
END;
/
```

10. **Display how many years has each cat using a varray. Handle the exception that may occur when a cat has an invalid age.**

Solution:

```
DECLARE
e_age EXCEPTION;
PRAGMA EXCEPTION_INIT(e_age, -20797);

  TYPE t_rec_age IS RECORD(
  cat_name proj_cats.cat_name%TYPE,
  age proj_cats.age%TYPE);

  TYPE t_ibt IS VARRAY(300) OF t_rec_age;
  t t_ibt;
  i PLS_INTEGER;
BEGIN
 SELECT cat_name, age BULK COLLECT INTO t
   FROM proj_cats ORDER BY age;

  DBMS_OUTPUT.PUT_LINE('Number of cats: ' || t.COUNT);

  i:=t.FIRST;
  WHILE i IS NOT NULL LOOP
     IF t(i).age = 0 THEN
     RAISE_APPLICATION_ERROR(-20797, 'Invalid cat age.');
     ELSIF t(i).age = 1 THEN
     DBMS_OUTPUT.PUT_LINE(i||'.'||t(i).cat_name||' has one year.');
     ELSE
     DBMS_OUTPUT.PUT_LINE(i||'.'||t(i).cat_name||' has '||t(i).age||' years.');
     END IF;
     i:=t.NEXT(i);
  END LOOP;

EXCEPTION
WHEN e_age THEN
DBMS_OUTPUT.PUT_LINE(SQLERRM);

END;
/
```

11. **Display how many cats have been to other contests using a varray. Handle the exceptions that may occur when we do not know if a cat has been or has not been to another contest.**

Solution:

```
DECLARE
e_contest EXCEPTION;
PRAGMA EXCEPTION_INIT(e_contest, -20727);

  TYPE t_rec_contest IS RECORD(
  cat_name proj_cats.cat_name%TYPE,
  other_contests proj_cats.other_contests%TYPE);

  TYPE t_ibt IS VARRAY(300) OF t_rec_contest;
  t t_ibt;
  i PLS_INTEGER;
BEGIN
 SELECT cat_name, other_contests BULK COLLECT INTO t
   FROM proj_cats ORDER BY cat_name;

  i:=t.FIRST;
  WHILE i IS NOT NULL LOOP
    IF t(i).other_contests = 'NO' THEN
    DBMS_OUTPUT.PUT_LINE(i||'-->'||t(i).cat_name||' has NOT been to other contests.');
    ELSIF t(i).other_contests = 'YES' THEN
    DBMS_OUTPUT.PUT_LINE(i||'-->'||t(i).cat_name||' has been to other contests.');
    ELSE
    RAISE_APPLICATION_ERROR(-20797, 'We do not know this information about this cat.');
    END IF;
    i:=t.NEXT(i);
  END LOOP;

EXCEPTION
WHEN e_contest THEN
DBMS_OUTPUT.PUT_LINE(SQLERRM);

END;
/
```

12. **Display the cat from the same country as the winner. If there are more cats from that country, handle the exception.**

Solution:

```
DECLARE
v_name proj_cats.cat_name%TYPE;
BEGIN
  SELECT cat_name INTO v_name FROM proj_cats c, proj_owners o, proj_addresses a WHERE
c.owner_id = o.owner_id AND o.address_id = a.address_id AND country_name =
                    (SELECT country_name FROM proj_addresses a, proj_owners o, proj_cats c,
proj_places p
                    WHERE a.address_id = o.address_id AND o.owner_id = c.owner_id AND
c.cat_id = p.cat_id
                    AND p.place='1st');
DBMS_OUTPUT.PUT_LINE('The cats from the same country as the winner cat are: '||v_name||'.');

EXCEPTION
  WHEN TOO_MANY_ROWS THEN
  DBMS_OUTPUT.PUT_LINE('There are more participants from that country, consider using a
cursor.');
END;
/
```

13. **A judge checks if they have a Siamese cat in the contest. If there is no cat of that breed, handle the exception.**

Solution:

```
DECLARE
v_breed proj_cats.breed%TYPE := 'Siamese';
BEGIN
 SELECT breed INTO v_breed FROM proj_cats WHERE breed = v_breed;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('Cat breed '||v_breed||' cannot be found.');

END;
/
```

14. **Display a certain cat based on its age. For exemplification, display the cat which is 8 years old. Handle the exceptions that may occur.**

Solution:

```
DECLARE
v_age proj_cats.age%TYPE := 8;
BEGIN
SELECT age INTO v_age FROM proj_cats WHERE age = v_age;

EXCEPTION
  WHEN TOO_MANY_ROWS THEN
  DBMS_OUTPUT.PUT_LINE('There are multiple cats with '||v_age||' years old, consider using a
cursor.');
  WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('There is no cat with '||v_age||' years old.');
  WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Another type of error occurred.');

END;
/
```

15. **An owner inserts a new cat. If that cat name is already used in the contest, handle the exception.**

Solution:

```
DECLARE
e_insert_excep EXCEPTION;
PRAGMA EXCEPTION_INIT(e_insert_excep, -00001);
BEGIN
INSERT INTO proj_cats (cat_id, cat_name) VALUES (14, 'Muffin');
DBMS_OUTPUT.PUT_LINE('Rows updated: '||SQL%ROWCOUNT);

EXCEPTION
  WHEN e_insert_excep THEN
  DBMS_OUTPUT.PUT_LINE('We already have a cat with that name in our contest. Insert
failed.');
END;
/
```

16. **A judge tries to see a specific cat based on its years. If he/she tries to input a string instead of a number for the age, handle the exception.**

Solution:

```
DECLARE
e_invalid_number EXCEPTION;
PRAGMA EXCEPTION_INIT(e_invalid_number, -01722);
v_age proj_cats.age%TYPE;
v_birthdate proj_cats.birth_date%TYPE;
BEGIN
SELECT age, birth_date INTO v_age, v_birthdate FROM proj_cats WHERE age = 'two';
DBMS_OUTPUT.PUT_LINE('The cat with '||v_age||' years is born on '||v_birthdate||'.');

EXCEPTION
  WHEN e_invalid_number THEN
  DBMS_OUTPUT.PUT_LINE('You are trying to input a string value in a numeric column.');
END;
/
```

17. **Raise an exception when the user tries to update a judge based on an id that does not exist and handle it.**

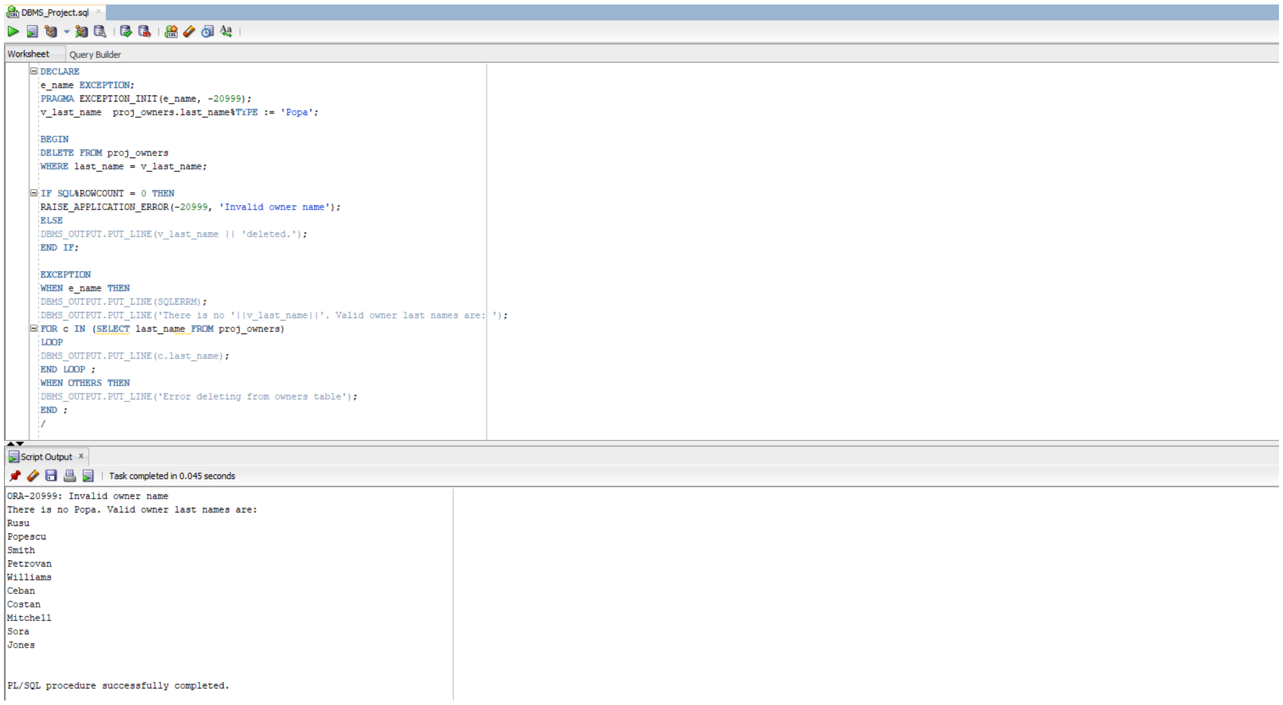
Solution:

```
DECLARE
e_invalid_id EXCEPTION;
v_judge_name  proj_judges.judge_name%TYPE := 'Sofia Vergara';
v_judge_id NUMBER := 21;
BEGIN
  UPDATE proj_judges
     SET judge_name = v_judge_name
     WHERE judge_id = v_judge_id;
  IF SQL%NOTFOUND THEN
     RAISE e_invalid_id;
  END IF;

EXCEPTION
WHEN e_invalid_id THEN
   DBMS_OUTPUT.PUT_LINE('No such judge id.');
END;
/
```

18. Handle the exceptions that may occur when someone tries to delete an owner based on a last name that does not exist in the database.

Solution:

```
DECLARE
e_name EXCEPTION;
PRAGMA EXCEPTION_INIT(e_name, -20999);
v_last_name  proj_owners.last_name%TYPE := 'Popa';

BEGIN
DELETE FROM proj_owners
WHERE last_name = v_last_name;

IF SQL%ROWCOUNT = 0 THEN
RAISE_APPLICATION_ERROR(-20999, 'Invalid owner name');
ELSE
DBMS_OUTPUT.PUT_LINE(v_last_name || 'deleted.');
END IF;

EXCEPTION
WHEN e_name THEN
DBMS_OUTPUT.PUT_LINE(SQLERRM);
DBMS_OUTPUT.PUT_LINE('There is no '||v_last_name||'. Valid owner last names are: ');
FOR c IN (SELECT last_name FROM proj_owners)
LOOP
DBMS_OUTPUT.PUT_LINE(c.last_name);
END LOOP ;
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error deleting from owners table');
END ;
/
```



```
DECLARE
e_name EXCEPTION;
PRAGMA EXCEPTION_INIT(e_name, -20999);
v_last_name  proj_owners.last_name%TYPE := 'Popa';

BEGIN
DELETE FROM proj_owners
WHERE last_name = v_last_name;

IF SQL%ROWCOUNT = 0 THEN
RAISE_APPLICATION_ERROR(-20999, 'Invalid owner name');
ELSE
DBMS_OUTPUT.PUT_LINE(v_last_name || 'deleted.');
END IF;

EXCEPTION
WHEN e_name THEN
DBMS_OUTPUT.PUT_LINE(SQLERRM);
DBMS_OUTPUT.PUT_LINE('There is no '||v_last_name||'. Valid owner last names are: ');
FOR c IN (SELECT last_name FROM proj_owners)
LOOP
DBMS_OUTPUT.PUT_LINE(c.last_name);
END LOOP ;
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error deleting from owners table');
END ;
/
```

```
ORA-20999: Invalid owner name
There is no Popa. Valid owner last names are:
Rusu
Popescu
Smith
Petrovan
Williams
Ceban
Costan
Mitchell
Sora
Jones


PL/SQL procedure successfully completed.
```

19. Display the manager of the judges. (the person that does not respond to anyone)

Solution:

```
DECLARE
v_manager proj_judges.judge_name%TYPE;
BEGIN
SELECT judge_name INTO v_manager FROM proj_judges WHERE manager_id IS NULL;
DBMS_OUTPUT.PUT_LINE('The manager is: '||v_manager||'.');
END;
/
```

20. There is only one Devon Rex in the contest, display its name.

Solution:

```
DECLARE
v_name proj_cats.cat_name%TYPE;
BEGIN
SELECT cat_name INTO v_name FROM proj_cats WHERE breed LIKE '%Rex';
DBMS_OUTPUT.PUT_LINE('The Devon Rex is named '||v_name||'.');
END;
/
```

21. Display the predominant type of fur and how many cats have that type of fur.

Solution:

```
DECLARE
CURSOR cur_fur IS SELECT fur_type, COUNT(*) no_of_fur FROM proj_cats
        GROUP BY fur_type
        ORDER BY no_of_fur DESC
        FETCH FIRST 1 ROW ONLY;
v_fur_record cur_fur%ROWTYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE('The predominant type of fur is:');
OPEN cur_fur;
  LOOP
   FETCH cur_fur INTO v_fur_record;
   EXIT WHEN cur_fur%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_fur_record.fur_type || ' - ' || v_fur_record.no_of_fur||' cats have
long fur');
  END LOOP;
CLOSE cur_fur;
END;
/
```

22. **Display the unique cat breeds in the contest.**

Solution:

```
DECLARE
CURSOR cur_breed IS SELECT breed, COUNT(*) no_of_breed FROM proj_cats
            HAVING COUNT(*) = 1
            GROUP BY breed
            ORDER BY no_of_breed;
v_breed_record cur_breed%ROWTYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE('The unique type of breed is:');
OPEN cur_breed;
  LOOP
   FETCH cur_breed INTO v_breed_record;
   EXIT WHEN cur_breed%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_breed_record.breed || ' - only ' || v_breed_record.no_of_breed || '
in the whole contest');
  END LOOP;
CLOSE cur_breed;
END;
/
```

23. **Display the judges that are Masters (have more than 5 years of experience) using an explicit cursor with parameters.**

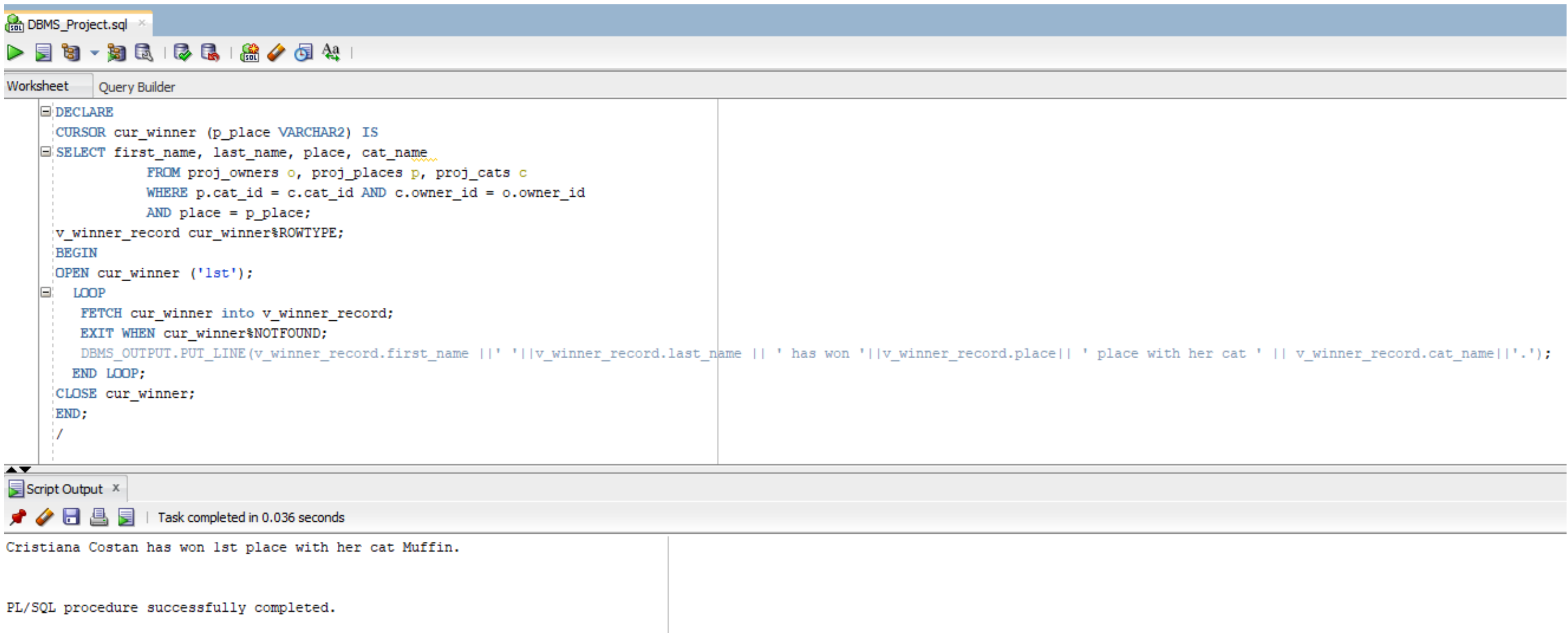
Solution:

```
DECLARE
CURSOR cur_judge (p_years NUMBER) IS
SELECT judge_name, years_of_experience FROM proj_judges
                      WHERE years_of_experience > p_years
                      ORDER BY years_of_experience DESC;
v_years NUMBER(2);
v_judge_record cur_judge%ROWTYPE;
BEGIN
v_years := 5;
DBMS_OUTPUT.PUT_LINE('The judges that are Masters are: ');
OPEN cur_judge (v_years);
  LOOP
   FETCH cur_judge into v_judge_record;
   EXIT WHEN cur_judge%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_judge_record.judge_name || ' - ' ||
v_judge_record.years_of_experience || ' years of experience');
  END LOOP;
CLOSE cur_judge;
END;
/
```

24. Display the person who got 1st place along with his/her cat.

Solution:

```
DECLARE
CURSOR cur_winner (p_place VARCHAR2) IS
SELECT first_name, last_name, place, cat_name
          FROM proj_owners o, proj_places p, proj_cats c
          WHERE p.cat_id = c.cat_id AND c.owner_id = o.owner_id
          AND place = p_place;
v_winner_record cur_winner%ROWTYPE;
BEGIN
OPEN cur_winner ('1st');
  LOOP
   FETCH cur_winner into v_winner_record;
   EXIT WHEN cur_winner%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_winner_record.first_name ||' '||v_winner_record.last_name || ' 
has won '||v_winner_record.place|| ' place with her cat ' || v_winner_record.cat_name||'.');
  END LOOP;
CLOSE cur_winner;
END;
/
```



```
DECLARE
CURSOR cur_winner (p_place VARCHAR2) IS
SELECT first_name, last_name, place, cat_name
          FROM proj_owners o, proj_places p, proj_cats c
          WHERE p.cat_id = c.cat_id AND c.owner_id = o.owner_id
          AND place = p_place;
v_winner_record cur_winner%ROWTYPE;
BEGIN
OPEN cur_winner ('1st');
  LOOP
   FETCH cur_winner into v_winner_record;
   EXIT WHEN cur_winner%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_winner_record.first_name ||' '||v_winner_record.last_name || ' has won '||v_winner_record.place|| ' place with her cat ' || v_winner_record.cat_name||'.');
  END LOOP;
CLOSE cur_winner;
END;
/
```

Task completed in 0.036 seconds

```
Cristiana Costan has won 1st place with her cat Muffin.


PL/SQL procedure successfully completed.
```

25. Display the cats by their sex.

Solution:

```
DECLARE
CURSOR cur_sex (p_sex VARCHAR2) IS
SELECT cat_name, sex FROM proj_cats
                WHERE sex = p_sex;
v_sex_record cur_sex%ROWTYPE;
BEGIN
OPEN cur_sex ('F');
  LOOP
   FETCH cur_sex into v_sex_record;
   EXIT WHEN cur_sex%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_sex_record.cat_name ||' is a female.');
  END LOOP;
CLOSE cur_sex;

OPEN cur_sex ('M');
  LOOP
   FETCH cur_sex into v_sex_record;
   EXIT WHEN cur_sex%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_sex_record.cat_name ||' is a male.');
  END LOOP;
CLOSE cur_sex;

END;
/
```

26. Display the cats classified by their size.

Solution:

```
DECLARE
CURSOR cur_size (p_size VARCHAR2) IS
SELECT cat_name, cat_size FROM proj_cats
           WHERE cat_size = p_size;
v_size_record cur_size%ROWTYPE;
BEGIN

OPEN cur_size ('small');
  LOOP
   FETCH cur_size into v_size_record;
   EXIT WHEN cur_size%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_size_record.cat_name ||' is a small-sized cat.');
  END LOOP;
```

```
CLOSE cur_size;

OPEN cur_size ('medium');
  LOOP
   FETCH cur_size into v_size_record;
   EXIT WHEN cur_size%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_size_record.cat_name ||' is a medium-sized cat.');
  END LOOP;
CLOSE cur_size;

OPEN cur_size ('large');
  LOOP
   FETCH cur_size into v_size_record;
   EXIT WHEN cur_size%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_size_record.cat_name ||' is a large-sized cat.');
  END LOOP;
CLOSE cur_size;

END;
/
```

27. **Create a table named "proj_contest" that will store information about the contest. If it already exists, handle the exception that may occur by deleting it and creating it again.**

Solution:

```
DECLARE
 e_table_exists exception;
 pragma exception_init(e_table_exists,-955);
BEGIN
  EXECUTE IMMEDIATE 'CREATE TABLE proj_contest (contest_id NUMBER, contest_date
DATE, contest_place VARCHAR2(20))';
    DBMS_OUTPUT.PUT_LINE('proj_contest has been created');
EXCEPTION
    WHEN e_table_exists THEN
       EXECUTE IMMEDIATE 'DROP TABLE proj_contest';
       EXECUTE IMMEDIATE 'CREATE TABLE proj_contest (contest_id NUMBER,
contest_date DATE, contest_place VARCHAR2(20))';
       DBMS_OUTPUT.PUT_LINE('proj_contest has been dropped and created again');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: '||SQLERRM);
END;
/
```

28. **Using execute immediate, display a cat's name based on a certain id.**

Solution:

```
DECLARE
v_statement VARCHAR2(100);
v_id NUMBER(4) := 12;
cat_rec proj_cats%ROWTYPE;
BEGIN
v_statement := 'SELECT * FROM proj_cats WHERE cat_id = :v_id';
EXECUTE IMMEDIATE v_statement INTO cat_rec USING v_id;
DBMS_OUTPUT.PUT_LINE('Cat id: ' || v_id);
DBMS_OUTPUT.PUT_LINE('Cat name: ' || cat_rec.cat_name);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: '||SQLERRM);
END;
/
```

29. **Create a procedure that displays how many different cat breeds are in the contest.**

Solution:

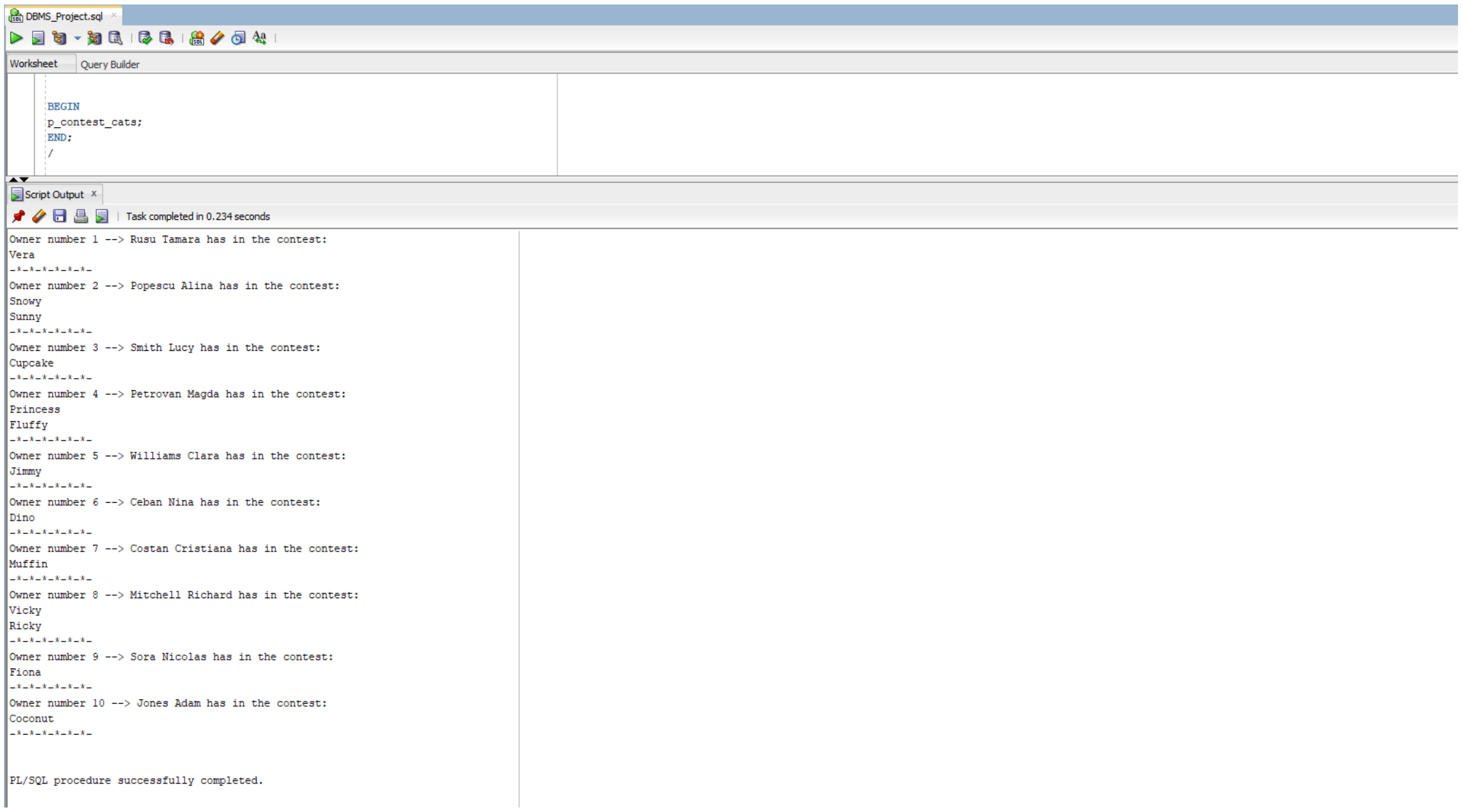
```
CREATE OR REPLACE PROCEDURE p_distinct_breeds IS
v_breeds proj_cats.breed%TYPE;
BEGIN
SELECT COUNT(DISTINCT breed) INTO v_breeds FROM proj_cats;
DBMS_OUTPUT.PUT_LINE('We have '||v_breeds||' distinct breeds in the contest.');
END p_distinct_breeds;
/

BEGIN
p_distinct_breeds;
END;
/
```

30. **Create a procedure that displays the owners and the name of each cat they brought in the contest.**

Solution:

```
CREATE OR REPLACE PROCEDURE p_contest_cats IS
CURSOR cur_owner IS SELECT DISTINCT o.owner_id, o.last_name || ' ' || o.first_name
owner_name FROM proj_cats c, proj_owners o
                WHERE c.owner_id = o.owner_id
                ORDER BY owner_id;
CURSOR cur_cat (p_owner_id NUMBER) IS SELECT o.owner_id, o.last_name || ' ' ||
o.first_name owner_name, c.cat_name FROM proj_cats c, proj_owners o
```

```
                    WHERE c.owner_id = o.owner_id AND o.owner_id = p_owner_id
                    ORDER BY owner_id;
v_owner cur_owner%ROWTYPE;
v_cat cur_cat%ROWTYPE;
BEGIN
OPEN cur_owner;
  LOOP
   FETCH cur_owner INTO v_owner;
   EXIT WHEN cur_owner%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Owner number '||v_owner.owner_id|| ' --> '
||v_owner.owner_name||' has in the contest:');
OPEN cur_cat (v_owner.owner_id);
  LOOP
   FETCH cur_cat INTO v_cat;
   EXIT WHEN cur_cat%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_cat.cat_name);
  END LOOP;
CLOSE cur_cat;
DBMS_OUTPUT.PUT_LINE('-*-*-*-*-*-*-');
  END LOOP;
CLOSE cur_owner;
END p_contest_cats;
/

BEGIN
p_contest_cats;
END;
/
```



```
BEGIN
p_contest_cats;
END;
/
```

Task completed in 0.234 seconds

```
Owner number 1 --> Rusu Tamara has in the contest:
Vera
-*-*-*-*-*-*-
Owner number 2 --> Popescu Alina has in the contest:
Snowy
Sunny
-*-*-*-*-*-*-
Owner number 3 --> Smith Lucy has in the contest:
Cupcake
-*-*-*-*-*-*-
Owner number 4 --> Petrovan Magda has in the contest:
Princess
Fluffy
-*-*-*-*-*-*-
Owner number 5 --> Williams Clara has in the contest:
Jimmy
-*-*-*-*-*-*-
Owner number 6 --> Ceban Nina has in the contest:
Dino
-*-*-*-*-*-*-
Owner number 7 --> Costan Cristiana has in the contest:
Muffin
-*-*-*-*-*-*-
Owner number 8 --> Mitchell Richard has in the contest:
Vicky
Ricky
-*-*-*-*-*-*-
Owner number 9 --> Sora Nicolas has in the contest:
Fiona
-*-*-*-*-*-*-
Owner number 10 --> Jones Adam has in the contest:
Coconut
-*-*-*-*-*-*-


PL/SQL procedure successfully completed.
```

31. Create a procedure that displays the junior (0-2 years) and senior (2+ years) cats.

Solution:

```
CREATE OR REPLACE PROCEDURE p_jun_sen IS
  CURSOR c_cat IS SELECT cat_name, age FROM proj_cats ORDER BY age;
  v_cat_record c_cat%ROWTYPE;
BEGIN
OPEN c_cat;
LOOP
FETCH c_cat INTO v_cat_record;
EXIT WHEN c_cat%NOTFOUND;
   IF v_cat_record.age BETWEEN 0 AND 2 THEN
   DBMS_OUTPUT.PUT_LINE(v_cat_record.cat_name||' is a junior cat.');
   ELSIF v_cat_record.age > 2 THEN
   DBMS_OUTPUT.PUT_LINE(v_cat_record.cat_name||' is a senior cat.');
  END IF;
END LOOP;
CLOSE c_cat;
END;
/

BEGIN
p_jun_sen;
END;
/
```

32. Create a procedure that displays a cat grade and place based on its id. Take id 7 for example.

Solution:

```
CREATE OR REPLACE PROCEDURE get_stats (p_id IN proj_places.cat_id%TYPE,
p_cat_grade OUT proj_places.grade%TYPE, p_cat_place OUT proj_places.place%TYPE) IS
BEGIN
  SELECT grade, place INTO p_cat_grade, p_cat_place
  FROM proj_places
  WHERE cat_id = p_id;
END get_stats;
/

DECLARE
  v_cat_grade proj_places.grade%TYPE;
  v_cat_place proj_places.place%TYPE;
BEGIN
  get_stats(7, v_cat_grade, v_cat_place);
  DBMS_OUTPUT.PUT_LINE('This cat got '||v_cat_place||' place with a grade of '||v_cat_grade);
   EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Invalid id');
END;
/
```

33. Create a procedure that displays the initials of each contestant.

Solution:

```
CREATE OR REPLACE PROCEDURE initials IS
CURSOR cur_initials IS
SELECT last_name, first_name, SUBSTR(last_name,0,1) || SUBSTR(first_name,0,1) AS initials
      FROM proj_owners
      ORDER BY SUBSTR(last_name,0,1);
v_initials_record cur_initials%ROWTYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE('The initials of each owner are: ');
OPEN cur_initials;
  LOOP
   FETCH cur_initials into v_initials_record;
   EXIT WHEN cur_initials%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(v_initials_record.last_name || ' ' || v_initials_record.first_name || ' -
' || v_initials_record.initials);
  END LOOP;
CLOSE cur_initials;
END initials;
/

BEGIN
  initials;
END;
/
```

34. Create a function to verify the validity of a judge id. Test it on judge number 18.

Solution:

```
CREATE OR REPLACE FUNCTION f_valid_id (p_judge_id proj_judges.judge_id%TYPE)
RETURN BOOLEAN IS
v_valid VARCHAR2(1);
BEGIN
SELECT 'x' INTO v_valid FROM proj_judges WHERE judge_id = p_judge_id;
RETURN(true);
EXCEPTION
  WHEN NO_DATA_FOUND THEN RETURN(false);
  WHEN OTHERS THEN NULL;
END;
/
```

```
BEGIN
IF f_valid_id(18) THEN
DBMS_OUTPUT.PUT_LINE('This judge id is valid.');
ELSE
DBMS_OUTPUT.PUT_LINE('This judge id is NOT valid.');
END IF;
END;
/
```

35. Create a function to display the age of the oldest cat in the contest.

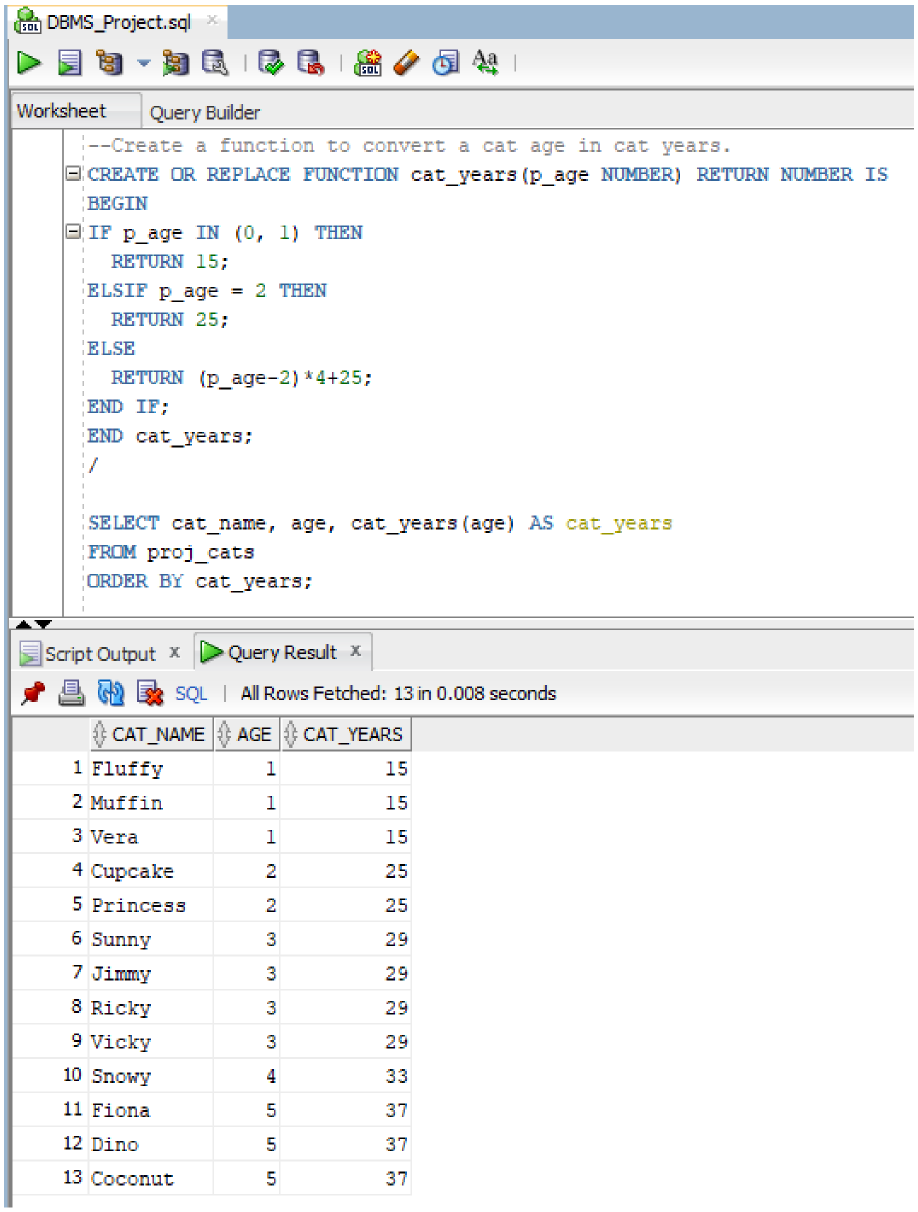
Solution:

```
CREATE OR REPLACE FUNCTION f_oldest_cat RETURN NUMBER IS
v_age proj_cats.age%TYPE;
BEGIN
SELECT age INTO v_age FROM proj_cats WHERE birth_date = (SELECT MIN(birth_date)
FROM proj_cats);
RETURN v_age;
END;
/

BEGIN
DBMS_OUTPUT.PUT_LINE('The oldest cat in the contest is about '||f_oldest_cat||' years old.');
END;
/
```

36. Create a function to convert a cat age in cat years.

Solution:

```
CREATE OR REPLACE FUNCTION cat_years(p_age NUMBER) RETURN NUMBER IS
BEGIN
IF p_age IN (0, 1) THEN
  RETURN 15;
ELSIF p_age = 2 THEN
  RETURN 25;
ELSE
  RETURN (p_age-2)*4+25;
END IF;
END cat_years;
/

SELECT cat_name, age, cat_years(age) AS cat_years
FROM proj_cats
ORDER BY cat_years;
```



```
--Create a function to convert a cat age in cat years.
CREATE OR REPLACE FUNCTION cat_years(p_age NUMBER) RETURN NUMBER IS
BEGIN
IF p_age IN (0, 1) THEN
  RETURN 15;
ELSIF p_age = 2 THEN
  RETURN 25;
ELSE
  RETURN (p_age-2)*4+25;
END IF;
END cat_years;
/

SELECT cat_name, age, cat_years(age) AS cat_years
FROM proj_cats
ORDER BY cat_years;
```

	CAT_NAME	AGE	CAT_YEARS
1	Fluffy	1	15
2	Muffin	1	15
3	Vera	1	15
4	Cupcake	2	25
5	Princess	2	25
6	Sunny	3	29
7	Jimmy	3	29
8	Ricky	3	29
9	Vicky	3	29
10	Snowy	4	33
11	Fiona	5	37
12	Dino	5	37
13	Coconut	5	37

37. Create a function to display if a cat's grade is below average.

Solution:

```
CREATE OR REPLACE FUNCTION cat_grades(p_grade NUMBER) RETURN VARCHAR2 IS
v_avg_grade proj_places.grade%TYPE;
BEGIN
SELECT AVG(grade) INTO v_avg_grade FROM proj_places;
IF p_grade < v_avg_grade THEN
  RETURN 'Lower than average';
ELSIF p_grade > v_avg_grade THEN
  RETURN 'Higher than average';
ELSE
```

```
  RETURN 'Unknown';
END IF;
END cat_grades;
/

SELECT cat_name, grade, cat_grades(grade) AS cat_grade
FROM proj_cats c, proj_places p
WHERE c.cat_id = p.cat_id
ORDER BY grade;
```

38. **Create a package that contains 2 public procedures, 1 private procedure and 2 public functions. The first function specifies if a cat got a grade higher than 8. The second function specifies which cat is the oldest and which one is the youngest. The first procedure shows the cats with odd IDs and the second procedure displays which award got each cat. The private procedure is invoked within the 2nd public one and returns the total number of cats in the contest.**

Solution:

```
CREATE OR REPLACE PACKAGE cats_pkg IS
FUNCTION higher_eight(p_cat_id IN proj_cats.cat_id%TYPE) RETURN VARCHAR2;
FUNCTION oldest_youngest(p_cat_id IN proj_cats.cat_id%TYPE) RETURN VARCHAR2;
PROCEDURE odd_ids;
PROCEDURE award_cats;
END;
/

CREATE OR REPLACE PACKAGE BODY cats_pkg IS
PROCEDURE count_cats(cats_no OUT NUMBER) IS
CURSOR cur_cats IS SELECT cat_id FROM proj_cats;
BEGIN
cats_no:=0;
FOR cat_rec IN cur_cats LOOP
SELECT COUNT(cat_id) INTO cats_no FROM proj_cats;
END LOOP;
END count_cats;

PROCEDURE odd_ids IS
v_cat_name proj_cats.cat_name%TYPE;
i NUMBER := 1;
BEGIN
dbms_output.put_line('The cats with even IDs are:');
WHILE i <= 13 LOOP
SELECT cat_name INTO v_cat_name FROM proj_cats WHERE cat_id=i;
dbms_output.put_line(i||'-->'||v_cat_name);
i:=i+2;
END LOOP;
END odd_ids;
```

```
PROCEDURE award_cats IS
v_diploma VARCHAR2(30);
  CURSOR c_place IS SELECT cat_name, place FROM proj_cats c, proj_places p WHERE
c.cat_id = p.cat_id ORDER BY grade DESC;
  v_place_record c_place%ROWTYPE;
  cats_no NUMBER(3);
BEGIN
count_cats(cats_no);
dbms_output.put_line('There are '||cats_no||' cats in the contest.');
OPEN c_place;
LOOP
FETCH c_place INTO v_place_record;
EXIT WHEN c_place%NOTFOUND;
  CASE
    WHEN v_place_record.place='1st' THEN v_diploma := 'Winner';
    WHEN v_place_record.place='2nd' THEN v_diploma := 'Second Place';
    WHEN v_place_record.place='3rd' THEN v_diploma := 'Third Place';
    ELSE v_diploma := 'Participation Diploma';
  END CASE;
  DBMS_OUTPUT.PUT_LINE(v_place_record.cat_name||' - '||v_diploma);
  END LOOP;
CLOSE c_place;
END award_cats;

FUNCTION higher_eight (p_cat_id IN proj_cats.cat_id%TYPE) RETURN VARCHAR2 IS
e_eight EXCEPTION;
PRAGMA EXCEPTION_INIT(e_eight, -20888);
v_grade proj_places.grade%TYPE;
BEGIN
SELECT grade INTO v_grade FROM proj_places WHERE cat_id = p_cat_id;
IF v_grade < 0 THEN
  RAISE_APPLICATION_ERROR(-20888, 'Invalid grade');
ELSIF v_grade > 8 THEN
  RETURN 'Yes';
ELSE
  RETURN 'No';
END IF;
EXCEPTION
WHEN e_eight THEN
DBMS_OUTPUT.PUT_LINE(SQLERRM);
END higher_eight;

FUNCTION oldest_youngest(p_cat_id IN proj_cats.cat_id%TYPE) RETURN VARCHAR2 IS
v_birth_date proj_cats.birth_date%TYPE;
v_max proj_cats.birth_date%TYPE;
v_min proj_cats.birth_date%TYPE;
BEGIN
```

```
SELECT birth_date INTO v_birth_date FROM proj_cats WHERE cat_id = p_cat_id;
SELECT birth_date INTO v_max FROM proj_cats WHERE birth_date = (SELECT
MAX(birth_date) FROM proj_cats);
SELECT birth_date INTO v_min FROM proj_cats WHERE birth_date = (SELECT
MIN(birth_date) FROM proj_cats);
IF v_birth_date = v_max THEN
  RETURN 'Oldest cat';
ELSIF v_birth_date = v_min THEN
  RETURN 'Youngest cat';
ELSE
  RETURN 'Not oldest, not youngest';
END IF;
END oldest_youngest;
END;
/

BEGIN
cats_pkg.odd_ids;
END;
/

BEGIN
cats_pkg.award_cats;
END;
/

SELECT cat_id, cat_name, cats_pkg.higher_eight(cat_id) AS "Grade higher than 8",
cats_pkg.oldest_youngest(cat_id) AS "Oldest/Youngest" FROM proj_cats ORDER BY birth_date;
```

39. Create a trigger to restrict the user from inserting or deleting a judge.

Solution:

```
CREATE OR REPLACE TRIGGER judges_table
  BEFORE INSERT OR DELETE ON proj_judges
BEGIN
IF INSERTING THEN
  RAISE_APPLICATION_ERROR(-20997, 'Inserting a new judge is forbidden!');
ELSIF DELETING THEN
  RAISE_APPLICATION_ERROR(-20998, 'Deleting a judge is forbidden!');
END IF;
END;
/

INSERT INTO proj_judges (judge_id, judge_name, cat_name, years_of_experience, manager_id)
values (20,'Marceline Vivienne',null,8,null);
```

```
DELETE FROM proj_judges
WHERE judge_id = 12;
```

40. **Create a trigger that keeps track of DML operations on the proj_addresses table into a logging table named log_addresses_table.**

Solution:

```
CREATE TABLE log_addresses_table (who_changed VARCHAR2(20), when_changed DATE);

CREATE OR REPLACE TRIGGER log_addresses_changes
AFTER INSERT OR UPDATE OR DELETE ON proj_addresses
BEGIN
  INSERT INTO log_addresses_table (who_changed, when_changed) VALUES (USER,
SYSDATE);
END;
/

INSERT INTO proj_addresses (address_id, country_id, country_name, city, street_name,
postal_code) values (77,'UK','United Kingdom','Oxford','Observatory St 77','OX7 7EP');

DELETE FROM proj_addresses WHERE address_id = 77;

SELECT * FROM log_addresses_table;
```

41. **Create a trigger to restrict the user from updating an owner last name on Saturdays or Sundays.**

Solution:

```
CREATE OR REPLACE TRIGGER update_wed_fri
  BEFORE UPDATE ON proj_owners
BEGIN
IF UPDATING('last_name') THEN
  IF TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
  RAISE_APPLICATION_ERROR (-20759, 'City Hall is closed, how did she/he manage to change
her/his name?');
  END IF;
END IF;
END;
/

UPDATE proj_owners
SET last_name = 'Sora'
WHERE owner_id = 7;

SELECT * FROM proj_owners;
```

42. **Create a trigger that prevents cats to have grades bigger than 10.**

Solution:

```
CREATE OR REPLACE TRIGGER reistrict_grade
  BEFORE INSERT OR UPDATE OF grade ON proj_places FOR EACH ROW
BEGIN
  IF :NEW.grade > 10 THEN
  RAISE_APPLICATION_ERROR (-20777,'Cat '||:NEW.cat_id||' cannot have a grade bigger than
10.');
  END IF;
END;
/

INSERT INTO proj_places (place, grade, cat_id) values ('ccc',77,77);
```

43. **Create a trigger that prevents decreasing the years of experience of judges.**

Solution:

```
CREATE OR REPLACE TRIGGER restrict_experience
  AFTER UPDATE OF years_of_experience ON proj_judges FOR EACH ROW
  WHEN (NEW.years_of_experience < OLD.years_of_experience)
BEGIN
  RAISE_APPLICATION_ERROR(-20979, 'Cannot decrease the years of experience!');
END;
/

UPDATE proj_judges
SET years_of_experience = 5
WHERE judge_id = 2;
```

44. **Create a trigger that specifies the old phone number along with the new one whenever a phone is updated.**

Solution:

```
CREATE OR REPLACE TRIGGER change_phone
  AFTER UPDATE OF phone_number ON proj_owners FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('The phone changed from '||:OLD.phone_number||' to
'||:NEW.phone_number||'.');
END;
/

UPDATE proj_owners
SET phone_number = '+77 70 7777 0777'
```

```
WHERE owner_id = 7;

UPDATE proj_owners
SET phone_number = '+4 076 422 1752'
WHERE owner_id = 7;
```