SULEYMAN DEMIREL UNIVERSITY
ENGINEERING FACULTY

CSS 305 - Database Management Systems 2

Project documentation

Checked by: Ulzhan Bissarinova

Made by: Alibek Aidarov

Aslan Kakashov

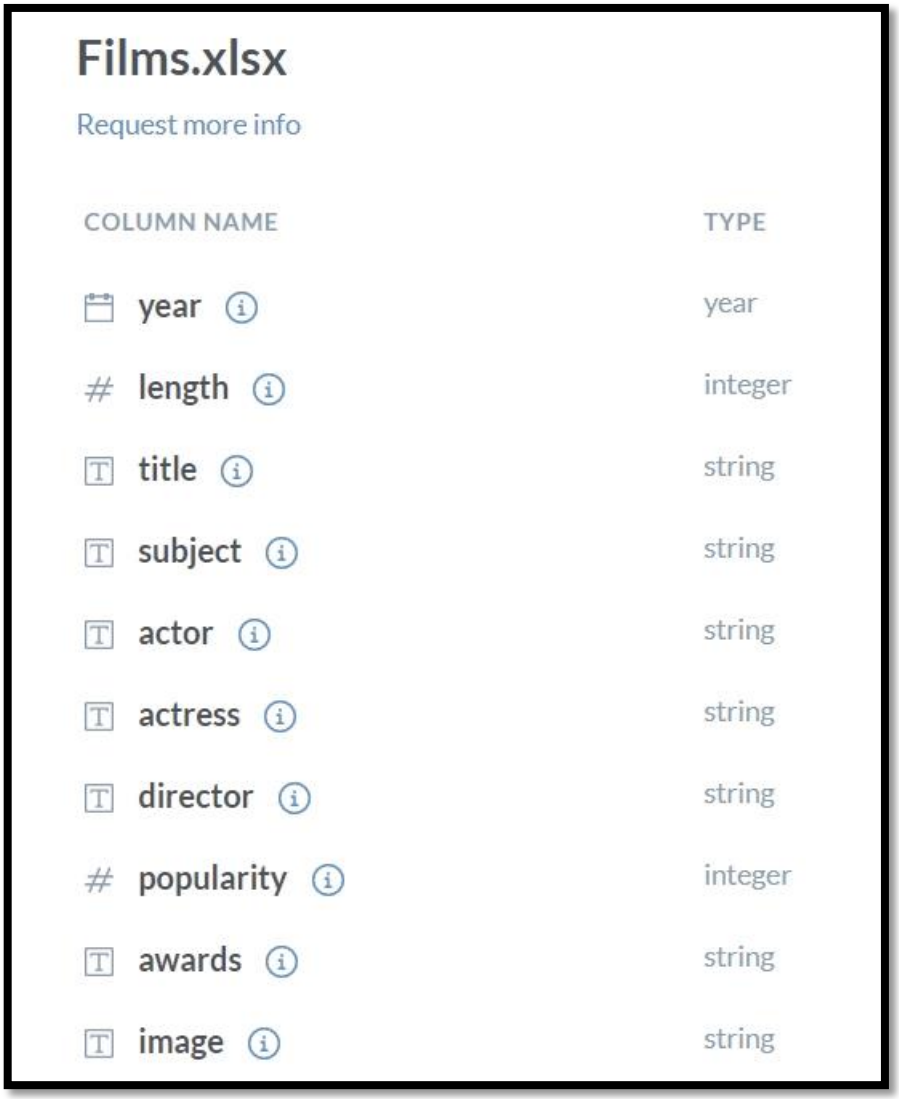
Illana Baitanatova

Temerlan Bakyt

Kaskelen, 2021

TASK 1

We chose this dataset : https://data.world/bramwax/movies/workspace/file?filename=Films.xlsx. This dataset is about films and consists of 10 columns.



Films.xlsx

Request more info

COLUMN NAME	TYPE
year	year
# length	integer
title	string
subject	string
actor	string
actress	string
director	string
# popularity	integer
awards	string
image	string

Figure 1.0

The type of “year” column is DATE , and “length” and “popularity” columns are numeric. And other columns are strings. In addition, this dataset contains about 1500 rows.

TASK 2

First, we imported the dataset into oracle.

Worksheet | Query Builder

Query Result

SQL | Fetched 50 rows in 0.004 seconds

	YEAR	LENGTH	TITLE	SUBJECT	ACTOR	ACTRESS	DIRECTOR	POPULARITY	AWARDS	IMAGE	ID	COUNT_OF_SEACRH
1	1989	108	Shirley Valentine	Comedy	Conti, Tom	Collins, Pauline	Gilbert, Lewis	51	No	NicholasCage.png	1	0
2	1992	135	City of Joy	Drama	Swayze, Patrick	Collins, Pauline	Joffe, Roland	87	No	NicholasCage.png	2	0
3	1966	99	Appaloosa, The	Western	Brando, Marlon	Comer, Anjanette	Furie, Sidney J.	15	No	brando.png	3	0
4	1986	88	Seven Minutes in Heaven	Comedy	Thames, Byron	Connelly, Jennifer	Feferman, Linda	49	No	NicholasCage.png	4	0
5	1991	96	Hearts of Darkness, A Filmmaker's Apocalypse	Drama	Bottoms, Sam	Coppola, Eleanor	Bahr, Fax	72	No	NicholasCage.png	5	0
6	1961	66	Tonight for Sure	Comedy	Lee, Karla	Cornell, Laura	Coppola, Francis Ford	4	No	NicholasCage.png	6	0
7	1990	110	White Hunter, Black Heart	Adventure	Eastwood, Clint	Cornwell, Charlotte	Eastwood, Clint	66	No	clintEastwood.png	7	0
8	1962	110	Sundays & Cybele	Drama	Kruger, Hardy	Courcel, Nicole	Bourguignon, Serge	11	Yes	NicholasCage.png	8	0
9	1989	90	Puppet Master	Science Fiction	LeMat, Paul	Crampton, Barbara	Schmoeller, David	20	No	NicholasCage.png	9	0
10	1991	95	Night Gallery	Horror	McDowall, Roddy	Crawford, Joan	Spielberg, Steven	31	No	NicholasCage.png	10	0
11	1989	103	Pet Sematary	Horror	Gwynne, Fred	Crosby, Denise	Lambert, Mary	27	No	NicholasCage.png	11	0
12	1992	60	America's Music, Gospel	Music	Phipps, Wentley	Crouch, Sandra	Walton, Kip	13	No	NicholasCage.png	12	0
13	1977	123	Slap Shot	Comedy	Newman, Paul	Crouse, Lindsay	Hill, George Roy	82	No	paulNewman.png	13	0
14	1987	109	O. C. & Stiggs	Comedy	Jenkins, Daniel H.	Curtin, Jane	Altman, Robert	3	No	NicholasCage.png	14	0
15	1988	108	A Fish Called Wanda	Comedy	Cleese, John	Curtis, Jamie Lee	Crichton, Charles	7	Yes	NicholasCage.png	15	0
16	1954	96	A Lesson in Love	Comedy	Björnstrand, Gunnar	Dahlbeck, Eva	Bergman, Ingmar	48	No	Bergman.png	16	0
17	1957	82	Brink of Life	Drama	Josephson, Erland	Dahlbeck, Eva	Bergman, Ingmar	57	No	Bergman.png	17	0
18	1986	120	Betty Blue	Drama	Anglade, Jean-Hughes	Dalle, Béatrice	Beineix, Jean-Jacques	71	No	NicholasCage.png	18	0
19	1979	122	Hair	Music	Savage, John	D'Angelo, Beverly	Forman, Milos	67	No	NicholasCage.png	19	0
20	1989	97	National Lampoon's Christmas Vacation	Comedy	Chase, Chevy	D'Angelo, Beverly	S, Jeremiah	81	No	NicholasCage.png	20	0
21	1974	124	Dersu Uzala, (The Hunter)	Adventure	Solomin, Yuri	Danilchenko, Svetlana	Kurosawa, Akira	81	Yes	NicholasCage.png	21	0
22	1990	106	Alice	Comedy	Baldwin, Alec	Danner, Blythe	Allen, Woody	22	No	woody.png	22	0
23	1980	90	Fifth Floor, The	Mystery	Hopkins, Bo	D'Arbanville, Patti	Avedis, Howard Hikmet	74	No	NicholasCage.png	23	0
24	1990	94	Snow Kill	Drama	Knox, Terence	D'Arbanville, Patti	Wright, Thomas J.	35	No	NicholasCage.png	24	0
25	1971	74	People, The	Drama	Shatner, William	Darby, Kim	Coppola, Francis Ford	36	No	NicholasCage.png	25	0
26	1969	128	True Grit	Western	Wayne, John	Darby, Kim	Hathaway, Henry	77	Yes	johnWayne.png	26	0
27	1942	18	Battle of Midway, The	War	Crisp, Donald	Darwell, Jane	Ford, John	75	No	johnFord.png	27	0
28	1948	103	Three Godfathers	Western	Wayne, John	Darwell, Jane	Ford, John	72	No	johnWayne.png	28	0
29	1965	133	Hush, Hush, Sweet Charlotte	Mystery	Cotten, Joseph	Davis, Bette	Aldrich, Robert	68	No	NicholasCage.png	29	0
30	1946	110	A Stolen Life	Drama	Ford, Glenn	Davis, Bette	Bernhardt, Curtis	20	No	glennFord.png	30	0
31	1939	96	Old Maid, The	Drama	Brent, George	Davis, Bette	Goulding, Edmund	18	No	NicholasCage.png	31	0
32	1950	138	All about Eve	Drama	Sanders, George	Davis, Bette	Mankiewicz, Joseph L.	23	Yes	NicholasCage.png	32	0
33	1986	96	Fly, The	Horror	Goldblum, Jeff	Davis, Geena	Cronenberg, David	33	No	NicholasCage.png	33	0

Figure 2.0 Dataset in oracle

We also added the "id" column, and made it as the primary key.

```
--triggers

ALTER TABLE  films  ADD (id NUMBER(10));
UPDATE  films  SET id=ROWNUM;
ALTER TABLE films MODIFY (id PRIMARY KEY not null);

CREATE SEQUENCE film_id START WITH 1911;

CREATE OR REPLACE TRIGGER film_id_add
    BEFORE INSERT ON films
    FOR EACH ROW
    BEGIN
      SELECT film_id.NEXTVAL
      INTO   :new.id
      FROM   dual;
END;
```

Figure 3.0

And we created a trigger to automatically fill in the new id. In addition, we have created a new table that stores the date of the administrator's action. (Admin can insert, delete, update films). In the picture below you can see our tables:

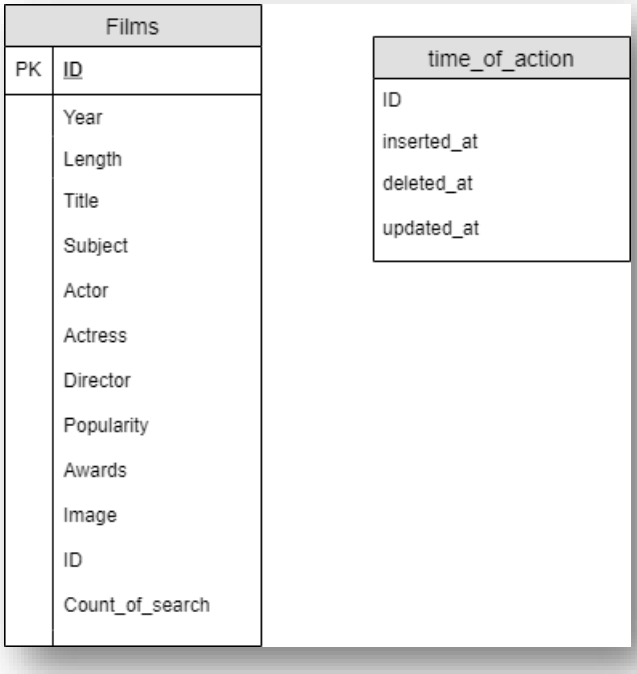


Figure 4.0

The type of "inserted_at", "updated_at" and "deleted_at" columns is DATE.

Also teacher, we have created 3 triggers to fill automatically "inserted_at", "updated_at" and "deleted_at" columns, each time when admin inserts, updates or deletes some rows in the "films" table.

```
CREATE OR REPLACE TRIGGER upd_film
AFTER UPDATE ON films FOR EACH ROW
BEGIN
    INSERT INTO time_of_actions(id, "updated_at")
    VALUES (:OLD.id , SYSDATE);
END;

CREATE OR REPLACE TRIGGER ins_film
AFTER INSERT ON films FOR EACH ROW
BEGIN
    INSERT INTO time_of_actions(id, "inserted_at")
    VALUES (:NEW.id , SYSDATE);
END;

CREATE OR REPLACE TRIGGER del_film
AFTER DELETE ON films FOR EACH ROW
BEGIN
    INSERT INTO time_of_actions(id, "deleted_at")
    VALUES (:OLD.id , SYSDATE);
END;
```

Figure 5.0 Our triggers.

Teacher, we created functions, packages, procedures, cursors, records and also used dynamic sql. We pushed all the code into the github. You can see the entire code at the link below.

https://github.com/gabdilqaq/final_project_DBMS2/blob/main/final.sql

TASK 3

We have created a website for this database. The frontend was written in html+css and the backend was written in python+cx_oracle+Flask.

(Flusk is a web framework. It offers useful tools and features to facilitate the process of creating web applications using Python.

cx_oracle is also a framework that allows you to connect web applications to oracle.)

And made a connection with oracle:

```
main.py   logs.html   index.html   logs.css   style.css
from flask import Flask, render_template, url_for, request, redirect
from werkzeug.utils import secure_filename
import cx_Oracle
import config
import os

cx_Oracle.init_oracle_client(lib_dir="/Users/Alibek/Downloads/instantclient_19_8")
connection = cx_Oracle.connect(config.username, config.password, config.dsn, encoding=config.encoding)

app = Flask(__name__)
```

Figure 6.0

```
main.py   config.py   logs.html
username = 'hr'
password = 'oracle'
dsn = 'localhost/orcl'
port = 1522
encoding = 'UTF-8'
```

Figure 7.0

So the main page of the website looks like this:

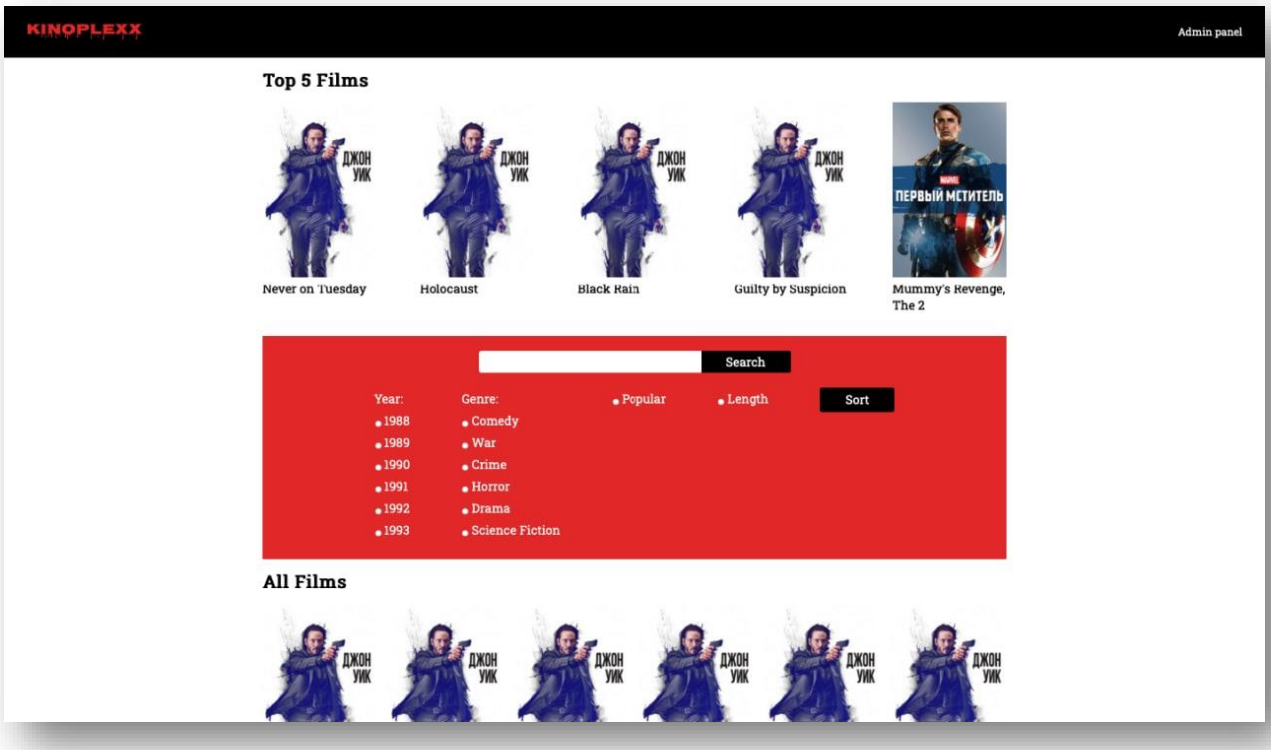


Figure 8.0 Our website.

In this page, the user can search for movies and sort them. We have created the "films_filter" package which contains functions to sort and to search the movies.

```
------------------------------------------------------------package filter
CREATE OR REPLACE PACKAGE films_filter AS

    FUNCTION length_desc RETURN t_table;
    FUNCTION popularity RETURN t_table;
    FUNCTION year_search(p_year VARCHAR2) RETURN t_table;
    FUNCTION subject(p_subject varchar2) RETURN t_table;
END;
```

Figure 9.0

Teacher you can see the body of package in github.

So as you can see our website gives opportunity to filter and to sort the films.

On top there is a recommendation system that offers the user top 5 films based on some algorithms. To do this, we created an "algos" package that contains a function and a procedure.

```
CREATE OR REPLACE PACKAGE algos AS
    PROCEDURE increment_count(p_id  IN films.ID%TYPE);
    FUNCTION top_five_film RETURN t_table;
END;
```

Figure 9.0

Teacher you can see the body of package in github.

The "increment_count" procedure increases the “count_of_search” column by one. So the “count_of_search” column initially is 0. And it will increase every time when user open this movie. And the “top_five_film” function returns 5 movies sorted by “count_of_search” column. We will demonstrate it in more detail when defending this project.

In “admin panel” admin can insert, update or delete films.

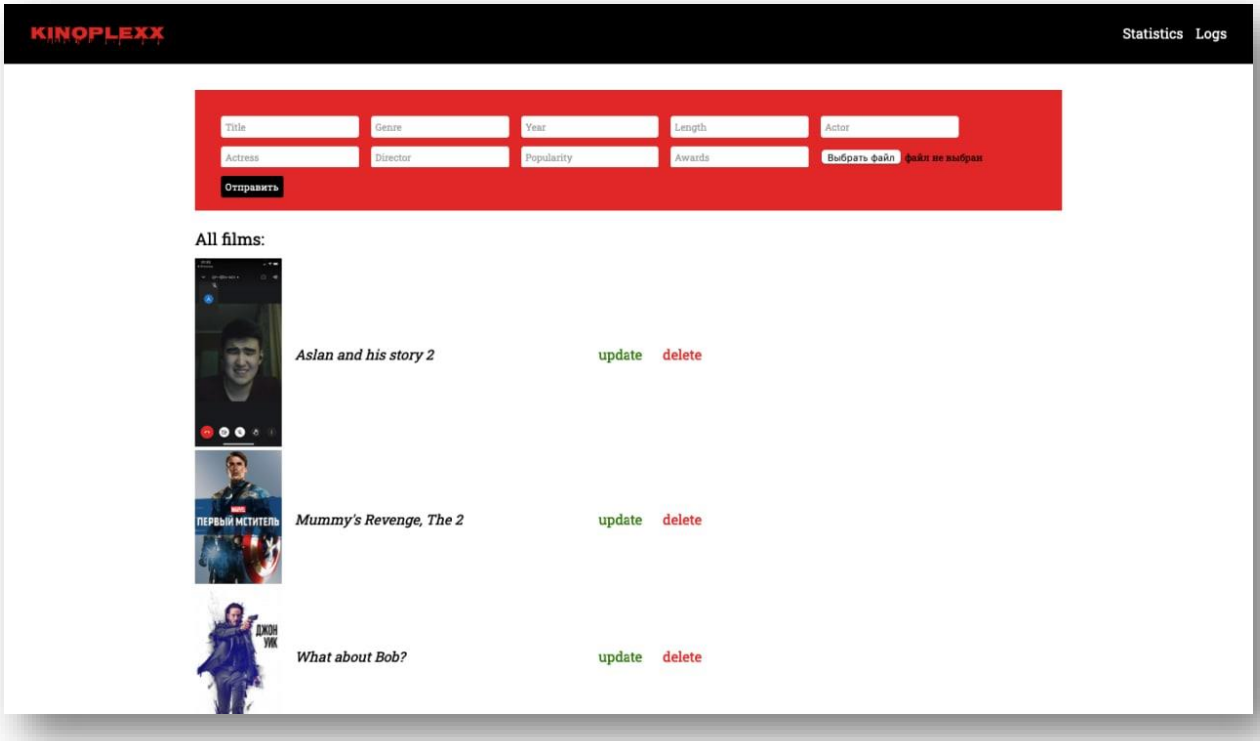


Figure 10.0

We created the “query” package to do this actions.

```
-------------------------------------------------------------package query
CREATE OR REPLACE PACKAGE query AS

    PROCEDURE insert_films(
    v_year IN films.year%TYPE,
    v_length IN films.length%TYPE,
    v_title IN films.title%TYPE,
    v_subject IN films.subject%TYPE,
    v_actor IN films.actor%TYPE,
    v_actress IN films.actress%TYPE,
    v_director IN films.director%TYPE,
    v_popularity IN films.popularity%TYPE,
    v_awards IN films.awards%TYPE,
    v_image IN films.image%TYPE);

    PROCEDURE update_films(
    v_year IN films.year%TYPE,
    v_length IN films.length%TYPE,
    v_title IN films.title%TYPE,
    v_subject IN films.subject%TYPE,
    v_actor IN films.actor%TYPE,
    v_actress IN films.actress%TYPE,
    v_director IN films.director%TYPE,
    v_popularity IN films.popularity%TYPE,
    v_awards IN films.awards%TYPE,
    v_image IN films.awards%TYPE,
    v_ID IN films.ID%TYPE);

    PROCEDURE delete_film(p_id IN films.ID%TYPE);
END;
```

Figure 11.0

Teacher, you can see the body of package in [github](). Inside of procedures in "query" package we used dynamic sql.

Also, as we said, we made 3 triggers that automatically fills the "time_of_actions" table, every time when admin inserts, updates or deletes the rows in "films" table. See the Figure 5.0.

KINOPLEXX Admin panel

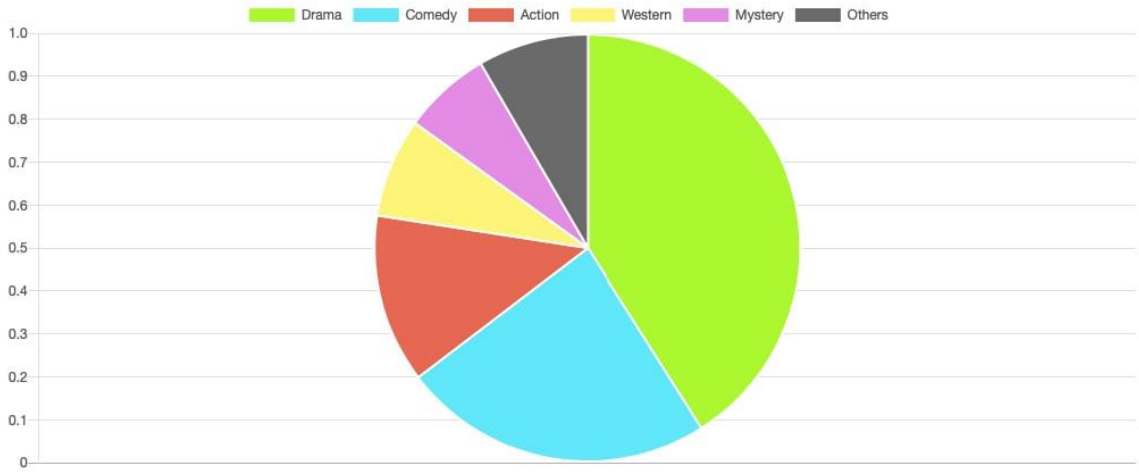
ID	Inserted_at	Updated_at	Deleted_at
1479	2021-05-02 07:35:15	None	None
1479	None	2021-05-02 07:35:29	None
1479	None	2021-05-02 07:36:32	None
1479	None	None	2021-05-02 07:36:55
1480	2021-05-02 08:59:59	None	None
1480	None	2021-05-02 12:32:13	None

Figure 12.0 The table “time_of_actions”

There are also two visual graphs on our website in the “statistics” page. The first one shows us the years with the highest release of films.

Figure 13.0

And the second shows the number of films by subject.



For this two charts we have created two functions:

```oracle_plsql
--FUNCTIONS FOR STATISTICS


CREATE OR REPLACE FUNCTION top_year
    RETURN t_top_year_table IS
    t_result t_top_year_table := t_top_year_table();
    v_i INTEGER:=1;
    CURSOR cur_top IS SELECT year , COUNT(*) AS "number_of_films"
            FROM films
            GROUP BY year
            ORDER BY COUNT(*) DESC;
BEGIN
    FOR v_rec_top IN cur_top LOOP
        t_result.extend;
            t_result(t_result.count) := t_top_year(null, null);
            t_result(t_result.count).year_of_release := v_rec_top.year;
            t_result(t_result.count).number_of_films :=  v_rec_top."number_of_films";
            EXIT WHEN v_i = 3;
            v_i:=v_i+1;
    END LOOP;
    RETURN t_result;
END;
```

Figure 14.0

```oracle_plsql
CREATE OR REPLACE FUNCTION top_subject
    RETURN t_top_subject_table IS
    t_result t_top_subject_table := t_top_subject_table();
    v_i INTEGER:=1;
    v_sum INTEGER := 0;
    v_percent NUMBER(10,2) ;
    v_percent_sum NUMBER(10,2):= 0;
    CURSOR cur_top IS SELECT subject , COUNT(*) AS "num_subject"
            FROM films
            GROUP BY subject
            ORDER BY COUNT(*) DESC;
BEGIN
    FOR v_rec_top IN cur_top LOOP
        v_sum := v_sum + v_rec_top."num_subject";
    END LOOP;

    FOR v_rec_top2 IN cur_top LOOP
        t_result.extend;
            t_result(t_result.count) := t_top_subjectt(null, null);
            t_result(t_result.count).subject := v_rec_top2.subject;
            v_percent := (v_rec_top2."num_subject"*100)/v_sum;
            v_percent_sum:=v_percent_sum+v_percent;
            t_result(t_result.count).percent_of_genre :=  v_percent;
            IF v_i = 6 THEN
                t_result(t_result.count).subject := 'Others';
                 t_result(t_result.count).percent_of_genre := (100-v_percent_sum);
            END IF;
            EXIT WHEN v_i = 6;
            v_i:=v_i+1;
    END LOOP;
    RETURN t_result;
END;
```

Figure 15.0

These two graphs depend on our data and are flexible.

Also teacher, we created function “all_films” to return all films in our database.

```
CREATE OR REPLACE FUNCTION all_films
    RETURN t_table IS
    t_result t_table :=t_table();
    CURSOR cur_all_film IS SELECT id ,title ,image FROM films;
BEGIN
    FOR v_rec_all_film IN cur_all_film LOOP
        t_result.extend;
            t_result(t_result.count) := t_films_table(null, null,null);
            t_result(t_result.count).id := v_rec_all_film.id;
            t_result(t_result.count).title :=  v_rec_all_film.title;
            t_result(t_result.count).image :=  v_rec_all_film.image;
    END LOOP;
    RETURN t_result;
END;
```

Figure 16.0

In addition, we created the “search_film " function, which searches for more than just one column. It searches for "title","actor"," actress "and" director " columns.

```
CREATE OR REPLACE FUNCTION search_film(p_film_name IN films.title%TYPE)
    RETURN t_table IS
    t_result t_table := t_table();
    CURSOR cur_title IS SELECT id,title,image FROM films
        WHERE (title LIKE '%'||p_film_name||'%' OR actor LIKE '%'||p_film_name||'%' OR
               actress LIKE '%'||p_film_name||'%' OR director LIKE '%'||p_film_name||'%')
               AND popularity IS NOT NULL
        ORDER BY popularity desc;
BEGIN
    FOR v_rec_film IN cur_title LOOP
        t_result.extend;
            t_result(t_result.count) := t_films_table(null, null,null);
            t_result(t_result.count).id := v_rec_film.id;
            t_result(t_result.count).title :=  v_rec_film.title;
            t_result(t_result.count).image :=  v_rec_film.image;
    END LOOP;
    RETURN t_result;
END;
```

Figure 17.0

In the end, we have 9 functions, 4 procedures, 4 triggers, 10 cursors, 3 dynamic sql, 4 tables , 1 sequence , 9 records.

So teacher, here is the link for frontend and backend side of our project:
https://github.com/gabdilqaq/final_project_DBMS2/tree/main/projectDB