Sistemas Informáticos 1, Práctica 2

Escuela Politécnica Superior, UAM

2022-2023

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

1. Introducción

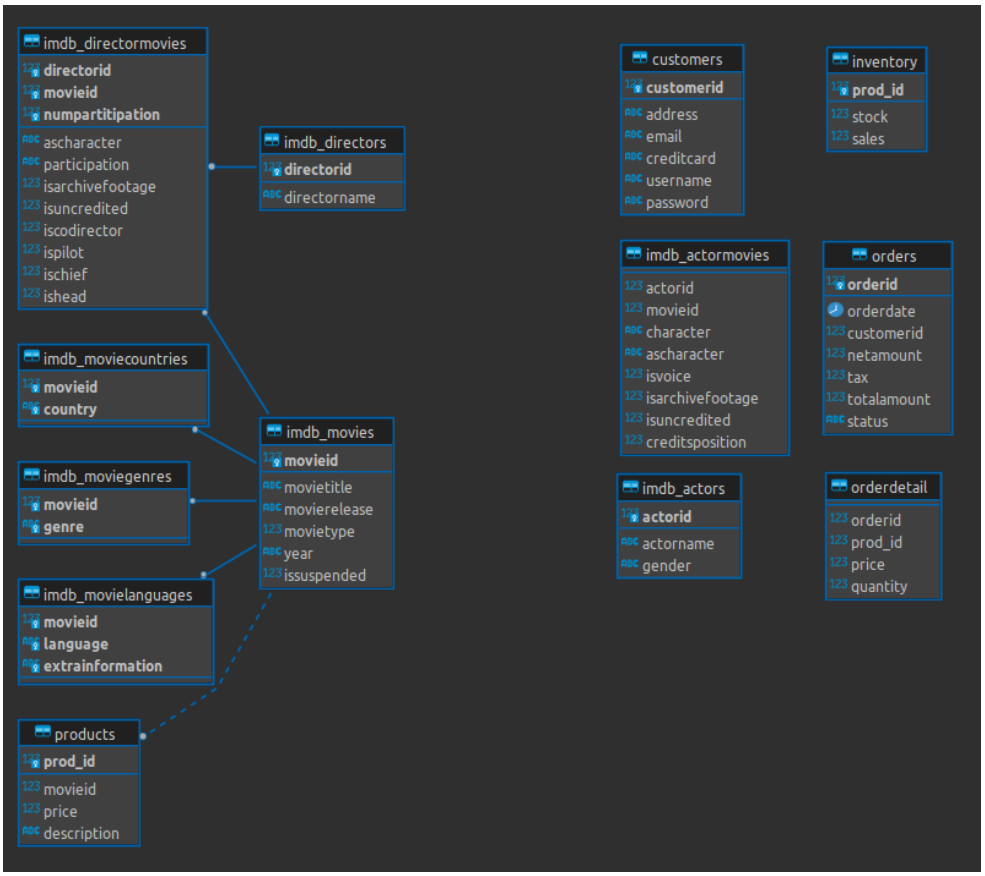
En esta práctica se trabaja con una base de datos para los elementos persistentes de forma dinámica en el sistema. En ella se almacenan datos como la información personal de los clientes, su saldo, historial de compras así como la relativa a las películas, actores…
El uso de la base de datos nos permite tener toda la información en un mismo lugar siendo accesibles y manipulables con todas las prestaciones y posibilidades que provee un sistema general de bases de datos.

El objetivo de la práctica consiste en entender conceptos de diseño, familiarizarse con el entorno de bases de datos, ser capaz de realizar consultas y acceder a base desde Python realizando actualizaciones y consultas bajo demanda del usuario.

2. Diseño de la BD

En primer lugar, analizamos la base de datos que se nos proporciona.

- Obtenemos el diagrama E-R mediante el uso de las herramientas de las que disponemos, en este caso, DBeaver.



Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

Como podemos observar, hay varios aspectos mejorables en la base que nos proporcionan; tales como las claves o los tipos de datos.

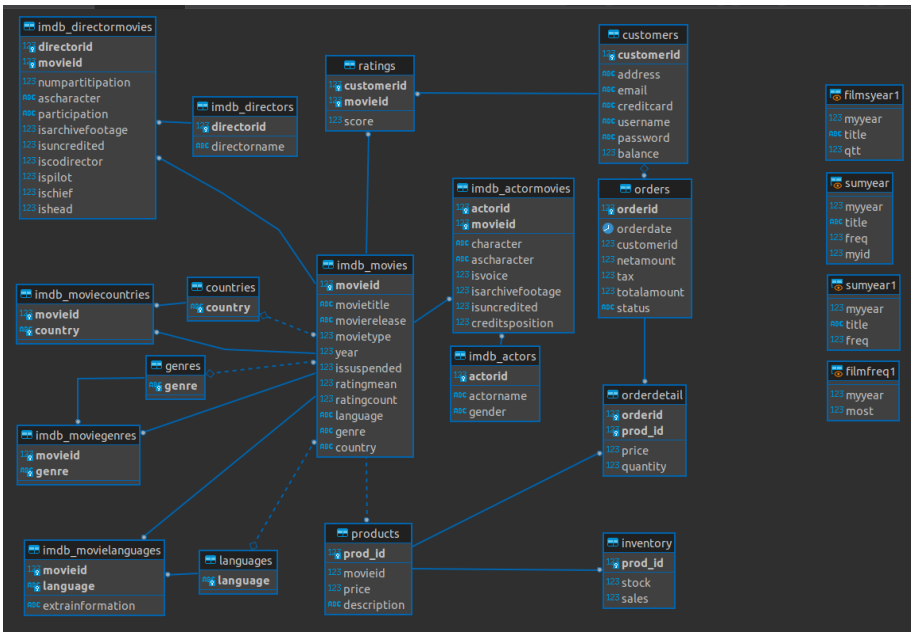
En primer lugar, nos fijamos en las **primary keys**: tanto la tabla de imdb_actormovies como la de orderdetail no tienen una clave primaria. Cosa que es esencial para identificar cada fila de la tabla.

Por otro lado, las **foreign keys**: vemos que todas las tablas de la derecha tienen atributos contenidos en otras tablas. Es necesario establecer estas claves foráneas como la relación de orders con customers, la de imdb_actormovies con imdb_actors y imdb_movies y la de orderdetail con orders e inventory.

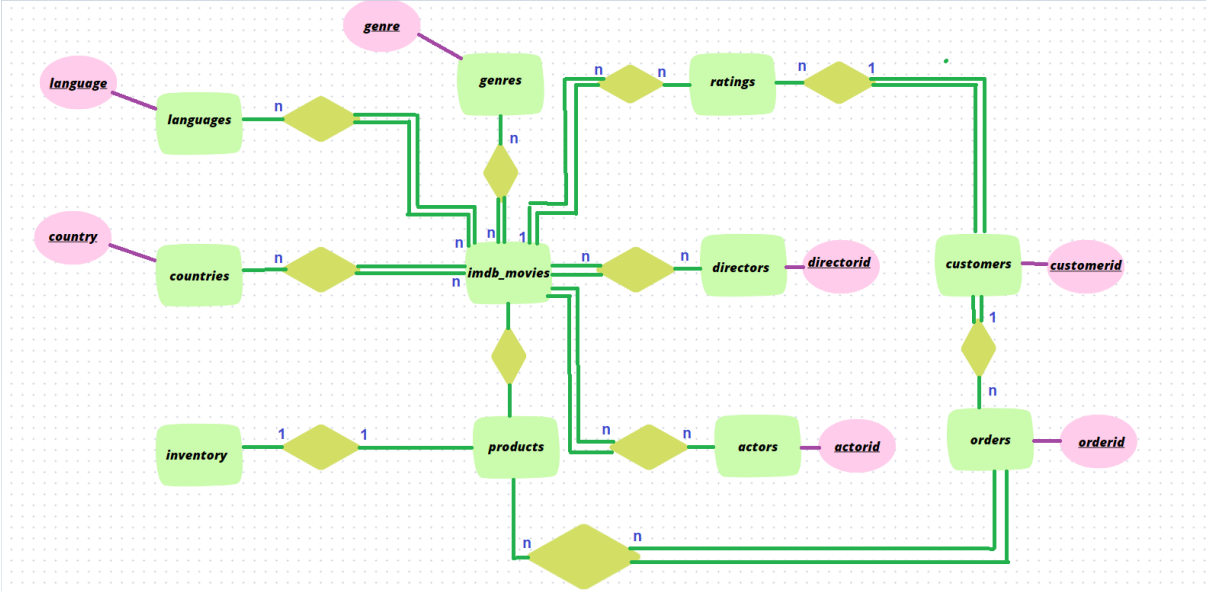
También es importante fijarse en el **tipo de dato** en función del uso que le vayamos a dar: por ejemplo, el atributo year de imdb_movies es texto cuando tiene más sentido que sea un entero de cuatro dígitos. También ocurre con el status en orders donde es más adecuado que las opciones sean una enumeración ya definida que texto.

Otro detalle a mencionar es la implementación de la relación imdb_movies con imdb_moviecountries, imdb_moviegenres e imdb_movielanguages, la cual se puede optimizar.

Diagrama entidad-relación de la base de datos resultante tras aplicar el script actualiza.sql y cambios introducidos, con su justificación.



Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08



Tras pasar el script de actualiza.sql y las funciones, hemos conseguido optimizar las relaciones entre las tablas mediante el uso de claves foráneas.
Hemos asignado claves primarias a aquellas que carecían de una y hemos corregido los formatos iniciales.
Por otro lado, hemos creado tres tablas para languages, countries y genres para así garantizar la integridad de los datos convirtiendo los atributos multivaludados.
También debido a las funciones de los apartados, aparecen nuevas vistas auxiliares que hemos utilizado para facilitar la ejecución de las consultas.

Análisis de la solución dada a las consultas, triggers y funciones solicitadas.

b) setPrice.sql

```
-- Sabiendo que los precios de las películas se han ido incrementando un 2% anualmente,
-- elaborar la consulta setPrice.sql que complete la columna 'price' de la tabla 'orderdetail',
-- sabiendo que el precio actual es el de la tabla 'products'.
UPDATE orderdetail
    SET price = aux.price
    FROM (SELECT products.price, orderdetail.orderid, orderdetail.prod_id
            FROM products, orderdetail
             WHERE products.prod_id = orderdetail.prod_id) AS aux
    WHERE (orderdetail.orderid = aux.orderid AND orderdetail.prod_id = aux.prod_id);
```

En esta función actualizamos el valor del atributo price en la tabla orderdetail de manera que coja el valor de price de products e identificamos los productos por su orderid y su prod_id.

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

c) setOrderAmount.sql

```
-- Una vez se disponga de esta información, realizar un procedimiento almacenado,
-- setOrderAmount, que complete las columnas 'netamount' (suma de los precios de las
-- películas del pedido) y 'totalamount' ('netamount' más impuestos) de la tabla 'orders'
-- cuando éstas no contengan ningún valor. Invocad de forma manual la consulta de
-- modificación setPrie y este procedimiento almacenado, para realizar una carga inicial

CREATE OR REPLACE FUNCTION setOrderAmount() returns void AS $$
BEGIN
    UPDATE orders
        SET netamount = aux.total,
            totalamount = aux.totalwt

        FROM (SELECT orders.orderid, sum(orderdetail.price*orderdetail.quantity) as total,
sum(orderdetail.price*orderdetail.quantity) + orders.tax as totalwt
            FROM orders, orderdetail
            WHERE orders.orderid = orderdetail.orderid
            GROUP BY orders.orderid) AS aux

        WHERE orders.orderid = aux.orderid;
END; $$
LANGUAGE 'plpgsql';

select setOrderAmount();
             WHERE products.prod_id = orderdetail.prod_id) AS aux
    WHERE (orderdetail.orderid = aux.orderid AND orderdetail.prod_id = aux.prod_id);
```

En esta función actualizamos el valor netamount de la tabla orders donde calculamos el total y el total + taxes en una tabla auxiliar con atributos total y totalwt respectivamente.

d) getTopSales.sql

La función POSTGRESQL que nos piden es la siguiente:

```
drop function if exists apartadoD(int,int);
CREATE OR REPLACE FUNCTION apartadoD(year1 INTEGER, year2 INTEGER) RETURNS TABLE(Year numeric, Film
varchar, Sales numeric) AS $$
BEGIN
    CREATE OR REPLACE VIEW filmsYear1 AS
        SELECT EXTRACT(year FROM orders.orderdate) as myYear, movies.movietitle as title, odt.quantity as
qtt
        FROM products, orders, orderdetail AS odt, imdb_movies AS movies
        WHERE products.prod_id = odt.prod_id AND products.movieid = movies.movieid AND odt.orderid =
orders.orderid
    ;
    CREATE OR REPLACE VIEW sumYear1 AS
        SELECT myYear, title, sum(qtt) as freq
        FROM filmsYear1
        GROUP BY myYear, title
    ;
    CREATE OR REPLACE VIEW filmFreq1 AS
        SELECT myYear, max(freq) AS most
        FROM sumYear1
        GROUP BY myYear
    ;

    RETURN QUERY
    SELECT sumYear1.myYear, sumYear1.title, filmFreq1.most
    FROM filmFreq1, sumYear1
    WHERE filmFreq1.myYear = sumYear1.myYear AND filmFreq1.most = sumYear1.freq AND filmFreq1.myYear >=
year1 AND filmFreq1.myYear <= year2
    ORDER BY filmFreq1.most DESC;

END;
$$ LANGUAGE plpgsql;

SELECT * FROM apartadoD(2015, 2018); WHERE products.prod_id = orderdetail.prod_id) AS aux
    WHERE (orderdetail.orderid = aux.orderid AND orderdetail.prod_id = aux.prod_id);
```

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

A esta función le llegan dos años como argumentos y devuelve una tabla mostrando el año, el título y el número total de ventas de la película con más ventas por cada año entre los dos años dados.
Sin embargo, a la hora de realizar la práctica nos dimos cuenta de que podríamos mejorarla para que fuera más eficiente de cara a nuestra página y a los criterios del apartado j). Por eso, esta función la entregamos como apartadoD.sql

e) getTopActors.sql

```
--Reciba un parámetro "género" y que devuelva los actores o actrices
--que más veces han actuado en dicho género, ordenados de
--más actuaciones a menos, siempre que hayan trabajado en más de 4 películas de ese
--género, con información de la película en la que debutaron para ese género, el año de esa
--película y quién (o quiénes) dirigió esa película (pueden aparecer varios registros para el
--mismo actor porque hizo varias películas su primer año, varios directores para la misma
--película, etc.)

drop function if exists topActors(varchar) cascade;

CREATE OR REPLACE FUNCTION topActors(g varchar) returns table(Actor VARCHAR, Num BIGINT, Debut INT, Film
VARCHAR, Director VARCHAR) AS $$
BEGIN
    --tabla con el número de películas de ese género que ha hecho cada actor (si es más de 4)
    create or replace view results AS
        select imdb_actors.actorname as actorname, count(*) as num
        from imdb_actors, imdb_actormovies, imdb_movies, imdb_moviegenres, imdb_directors,
imdb_directormovies
        where imdb_actors.actorid = imdb_actormovies.actorid
        and imdb_actormovies.movieid = imdb_movies.movieid
        and imdb_movies.movieid = imdb_moviegenres.movieid
        and imdb_moviegenres.genre = 'Action'
        and imdb_movies.movieid = imdb_directormovies.movieid
        and imdb_directormovies.directorid = imdb_directors.directorid
        group by imdb_actors.actorname
        having count(*)> 4
        order by count(*) desc
    ;

    return query
    select imdb_actors.actorname as actorname, results.num as num, imdb_movies.year as debut,
imdb_movies.movietitle as movietitle, imdb_directors.directorname as directorname
    from imdb_actors, imdb_movies, imdb_moviegenres, imdb_actormovies, imdb_directormovies,
imdb_directors, results
    where imdb_actors.actorid = imdb_actormovies.actorid
        and imdb_actormovies.movieid = imdb_movies.movieid
        and imdb_movies.movieid = imdb_moviegenres.movieid
        and imdb_movies.movieid = imdb_directormovies.movieid
        and imdb_directormovies.directorid = imdb_directors.directorid
        and results.actorname = imdb_actors.actorname
        and imdb_moviegenres.genre = 'Action'
    group by imdb_actors.actorname, imdb_movies.movietitle, imdb_directors.directorname,
imdb_movies.year, results.num
    order by results.num desc;
END;
$$ LANGUAGE plpgsql;

select * from topActors('Action');
```

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

En esta función lo primero que hemos hecho es una vista donde aparecen los actores junto al número de películas de ese género siempre y cuando sea más de dos.

A continuación, devolvemos la información que se nos pide (incluida la de la vista) ordenada de mayor a menor por el número de apariciones.

Evidencias de los resultados obtenidos.

b) setPrice.sql

Acciones		orderid	prod_id	price	quantity
Editar	Eliminar	1	1014	*NULL*	1
Editar	Eliminar	1	1288	*NULL*	1
Editar	Eliminar	1	1938	*NULL*	1
Editar	Eliminar	2	2443	*NULL*	1
Editar	Eliminar	2	3229	*NULL*	1
Editar	Eliminar	3	268	*NULL*	1
Editar	Eliminar	3	696	*NULL*	1
Editar	Eliminar	3	1467	*NULL*	1
Editar	Eliminar	3	1766	*NULL*	1
Editar	Eliminar	3	3215	*NULL*	1
Editar	Eliminar	3	3777	*NULL*	1
Editar	Eliminar	3	3802	*NULL*	1
Editar	Eliminar	3	4256	*NULL*	1
Editar	Eliminar	3	4505	*NULL*	1
Editar	Eliminar	3	4794	*NULL*	1
Editar	Eliminar	4	701	*NULL*	1
Editar	Eliminar	4	2159	*NULL*	1
Editar	Eliminar	4	4713	*NULL*	1
Editar	Eliminar	4	4795	*NULL*	1
Editar	Eliminar	4	5989	*NULL*	1
Editar	Eliminar	4	6181	*NULL*	1
Editar	Eliminar	4	6233	*NULL*	1
Editar	Eliminar	4	6627	*NULL*	1
Editar	Eliminar	5	254	*NULL*	1
Editar	Eliminar	5	1826	*NULL*	1
Editar	Eliminar	5	2984	*NULL*	1
Editar	Eliminar	5	3111	*NULL*	1
Editar	Eliminar	5	3399	*NULL*	1
Editar	Eliminar	5	4696	*NULL*	1
Editar	Eliminar	6	2996	*NULL*	1

Acciones		orderid	prod_id	price	quantity
Editar	Eliminar	3	4505	20.4	1
Editar	Eliminar	6	2996	15.6	1
Editar	Eliminar	11	4868	10	1
Editar	Eliminar	11	5821	15	2
Editar	Eliminar	16	6013	21.6	1
Editar	Eliminar	17	161	10	1
Editar	Eliminar	18	1425	13	1
Editar	Eliminar	18	2370	19.2	1
Editar	Eliminar	25	1188	10	1
Editar	Eliminar	27	690	17	1
Editar	Eliminar	28	3640	19	1
Editar	Eliminar	31	645	16.8	1
Editar	Eliminar	37	2018	13	1
Editar	Eliminar	38	6001	12	1
Editar	Eliminar	44	4092	14	1
Editar	Eliminar	46	1445	11	1
Editar	Eliminar	54	3282	15.6	1
Editar	Eliminar	59	2619	13.2	1
Editar	Eliminar	60	3016	14	1
Editar	Eliminar	61	2179	12	1
Editar	Eliminar	64	2327	11	1
Editar	Eliminar	65	5001	22.8	1
Editar	Eliminar	72	6059	17	1
Editar	Eliminar	83	4518	22.8	1
Editar	Eliminar	84	792	13	1
Editar	Eliminar	85	4716	17	1
Editar	Eliminar	93	1510	19.2	1
Editar	Eliminar	97	5068	15.6	1
Editar	Eliminar	98	3168	12	1
Editar	Eliminar	102	6568	19	1

Como podemos ver, la columna de precio pasa de ser NULL a tener el valor que le corresponde (de la tabla de productos).

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

c) setOrderAmount.sql

Acciones		orderid	orderdate	customerid	netamount	tax	totalamount	status
Editar	Eliminar	316	2021-06-30	17	NULL	15	NULL	Paid
Editar	Eliminar	1025	2021-03-25	71	NULL	15	NULL	Paid
Editar	Eliminar	6260	2019-02-21	455	NULL	15	NULL	Paid
Editar	Eliminar	7223	2017-01-04	529	NULL	15	NULL	Paid
Editar	Eliminar	7779	2018-01-25	565	NULL	15	NULL	Paid
Editar	Eliminar	8327	2018-08-29	595	NULL	15	NULL	Paid
Editar	Eliminar	8748	2019-11-20	626	NULL	15	NULL	Paid
Editar	Eliminar	9148	2022-02-22	658	NULL	18	NULL	Paid
Editar	Eliminar	9838	2020-02-26	723	NULL	15	NULL	Paid
Editar	Eliminar	12470	2019-06-23	936	NULL	15	NULL	Paid
Editar	Eliminar	12891	2022-03-18	965	NULL	18	NULL	Paid
Editar	Eliminar	14011	2021-09-10	1054	NULL	18	NULL	Paid
Editar	Eliminar	14815	2020-05-14	1111	NULL	15	NULL	Paid
Editar	Eliminar	15237	2017-06-05	1144	NULL	15	NULL	Paid
Editar	Eliminar	15642	2021-01-19	1177	NULL	15	NULL	Paid
Editar	Eliminar	16333	2019-11-05	1234	NULL	15	NULL	Paid
Editar	Eliminar	16754	2020-11-29	1262	NULL	15	NULL	Paid
Editar	Eliminar	17161	2018-08-13	1293	NULL	15	NULL	Paid
Editar	Eliminar	18126	2018-01-01	1369	NULL	15	NULL	Paid
Editar	Eliminar	18670	2018-10-05	1414	NULL	15	NULL	Paid
Editar	Eliminar	19220	2017-07-10	1462	NULL	15	NULL	Paid
Editar	Eliminar	20752	2018-11-22	1575	NULL	15	NULL	Paid
Editar	Eliminar	21027	2017-11-03	1596	NULL	15	NULL	Paid
Editar	Eliminar	22669	2018-10-14	1724	NULL	15	NULL	Paid
Editar	Eliminar	25032	2019-11-17	1909	NULL	15	NULL	Paid
Editar	Eliminar	27926	2020-05-12	2136	NULL	15	NULL	Paid
Editar	Eliminar	28328	2018-09-24	2166	NULL	15	NULL	Paid
Editar	Eliminar	29876	2018-10-04	2290	NULL	15	NULL	Paid
Editar	Eliminar	30115	2017-12-31	2310	NULL	15	NULL	Paid
Editar	Eliminar	31777	2019-05-04	2434	NULL	15	NULL	Paid

Acciones		orderid	orderdate	customerid	netamount	tax	totalamount	status
Editar	Eliminar	111777	2021-06-01	8639	136.8	15	151.8	Shipped
Editar	Eliminar	101140	2019-10-08	7809	169.6	15	184.6	Processed
Editar	Eliminar	96868	2018-03-04	7464	95.2	15	110.2	Shipped
Editar	Eliminar	72397	2017-07-31	5567	198.0	15	213.0	Shipped
Editar	Eliminar	56202	2018-08-16	4282	176.0	15	191.0	Shipped
Editar	Eliminar	30052	2020-04-04	2303	86.2	15	101.2	Shipped
Editar	Eliminar	106680	2020-12-21	8247	37	15	52	Processed
Editar	Eliminar	273	2020-05-08	15	128.1	15	143.1	Shipped
Editar	Eliminar	53335	2018-04-19	4056	133.9	15	148.9	Shipped
Editar	Eliminar	53426	2018-11-10	4063	108.4	15	123.4	Paid
Editar	Eliminar	28920	2020-06-03	2216	111.1	15	126.1	Processed
Editar	Eliminar	110395	2019-10-20	8524	75.0	15	90.0	Shipped
Editar	Eliminar	47237	2019-05-01	3589	84.4	15	99.4	Shipped
Editar	Eliminar	78549	2017-06-22	6033	109.6	15	124.6	Shipped
Editar	Eliminar	93623	2018-08-26	7215	122.2	15	137.2	Paid
Editar	Eliminar	84198	2017-04-30	6474	132.2	15	147.2	Processed
Editar	Eliminar	72774	2020-03-11	5596	93.9	15	108.9	Shipped
Editar	Eliminar	61351	2017-09-06	4696	193.4	15	208.4	Shipped
Editar	Eliminar	41733	2020-07-12	3167	119.1	15	134.1	Shipped
Editar	Eliminar	1750	2018-02-17	123	63.8	15	78.8	Processed
Editar	Eliminar	68926	2019-03-18	5292	184.9	15	199.9	Processed
Editar	Eliminar	67908	2018-08-30	5214	139.2	15	154.2	Shipped
Editar	Eliminar	24486	2017-10-10	1866	46.8	15	61.8	Shipped
Editar	Eliminar	4976	2017-07-06	363	92.6	15	107.6	Processed
Editar	Eliminar	29103	2022-01-09	2230	157.2	18	175.2	Shipped
Editar	Eliminar	79903	2017-06-19	6130	149.4	15	164.4	Paid
Editar	Eliminar	113452	2017-12-13	8776	88.2	15	103.2	Shipped
Editar	Eliminar	61392	2019-09-26	4698	153.9	15	168.9	Shipped
Editar	Eliminar	80421	2021-04-01	6168	118	15	133	Shipped
Editar	Eliminar	1552	2021-02-13	110	127.3	15	142.3	Processed

Al ejecutar esta función los valores de netamount y totalamount pasan de ser NULL a tener el valor que les corresponde.

d) getTopSales.sql

year	film	sales
2018	Impostors, The (1998)	135
2017	No Looking Back (1998)	105
2016	Porky's (1982)	15

getTopSales nos muestra la película más vendida en los años indicados y el número de ventas

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

e) getTopActors.sql

actor	num	debut	film	director
Welker, Frank	28	1997	Anaconda (1997)	Llosa, Luis
Welker, Frank	28	1999	Deep Blue Sea (1999)	Harlin, Renny
Welker, Frank	28	1998	Godzilla (1998)	Emmerich, Roland
Welker, Frank	28	1986	Golden Child, The (1986)	Ritchie, Michael (I)
Welker, Frank	28	1986	Great Mouse Detective, The (1986)	Clements, Ron
Welker, Frank	28	1986	Great Mouse Detective, The (1986)	Mattinson, Burny
Welker, Frank	28	1986	Great Mouse Detective, The (1986)	Michener, David
Welker, Frank	28	1986	Great Mouse Detective, The (1986)	Musker, John
Welker, Frank	28	1996	Independence Day (1996)	Emmerich, Roland
Welker, Frank	28	1995	Jumanji (1995)	Johnston, Joe (I)
Welker, Frank	28	2000	Road to El Dorado, The (2000)	Bergeron, Bibo
Welker, Frank	28	2000	Road to El Dorado, The (2000)	Finn, Will
Welker, Frank	28	2000	Road to El Dorado, The (2000)	Katzenberg, Jeffrey
Welker, Frank	28	2000	Road to El Dorado, The (2000)	Paul, Don (I)
Welker, Frank	28	2000	Road to El Dorado, The (2000)	Silverman, David (I)
Welker, Frank	28	1994	Shadow, The (1994)	Mulcahy, Russell
Welker, Frank	28	1997	Spawn (1997)	Dippé, Mark A.Z.
Welker, Frank	28	1995	Species (1995)	Donaldson, Roger
Welker, Frank	28	1984	Star Trek III: The Search for Spock (1984)	Nimoy, Leonard
Welker, Frank	28	1994	Stargate (1994)	Emmerich, Roland
Welker, Frank	28	1993	Super Mario Bros. (1993)	Jankel, Annabel
Welker, Frank	28	1993	Super Mario Bros. (1993)	Joffé, Roland
Welker, Frank	28	1993	Super Mario Bros. (1993)	Morton, Rocky
Welker, Frank	28	1993	Super Mario Bros. (1993)	Semler, Dean
Welker, Frank	28	1995	Tank Girl (1995)	Talalay, Rachel
Welker, Frank	28	1991	Teenage Mutant Ninja Turtles II: The Secret of the Ooze (1991)	Pressman, Michael
Welker, Frank	28	1986	Transformers: The Movie, The (1986)	Shin, Nelson (I)
Welker, Frank	28	1995	Virtuosity (1995)	Leonard, Brett (I)
Rosales Jr., Thomas	21	1994	Beverly Hills Cop III (1994)	Landis, John (I)
Rosales Jr., Thomas	21	1997	Con Air (1997)	West, Simon (I)
Rosales Jr., Thomas	21	1994	Crow, The (1994)	Proyas, Alex
Rosales Jr., Thomas	21	1997	Face/Off (1997)	Woo, John (I)
Rosales Jr., Thomas	21	1995	Heat (1995)	Mann, Michael (I)
Rosales Jr., Thomas	21	1993	Last Action Hero (1993)	McTiernan, John (I)
Rosales Jr., Thomas	21	1996	Last Man Standing (1996/I)	Hill, Walter (I)
Rosales Jr., Thomas	21	1997	Lost World: Jurassic Park, The (1997)	Spielberg, Steven (I)
Rosales Jr., Thomas	21	1994	Low Down Dirty Shame, A (1994)	Wayans, Keenen Ivory
Rosales Jr., Thomas	21	1981	Nighthawks (1981)	Malmuth, Bruce
Rosales Jr., Thomas	21	1981	Nighthawks (1981)	Nelson, Gary (I)
Rosales Jr., Thomas	21	1986	No Mercy (1986)	Pearce, Richard (I)
Rosales Jr., Thomas	21	1990	Predator 2 (1990)	Hopkins, Stephen
Rosales Jr., Thomas	21	1998	Replacement Killers, The (1998)	Fuqua, Antoine
Rosales Jr., Thomas	21	1990	RoboCop 2 (1990)	Kershner, Irvin
Rosales Jr., Thomas	21	1987	Running Man, The (1987)	Davis, Andrew (I)

…

Esta función muestra los actores que más participación tienen en un género dado. En este caso lo hemos ejecutado con ‘Action’. Mostrando primero el que tiene mayor número y la información de todas las películas.

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

Integración en el portal.

j) De cara a añadir a la página principal la tabla resultante en el apartado d) hicimos los siguientes cambios:

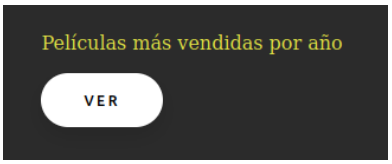
1. getTopSales.sql pasa a ser el siguiente:

```
drop function if exists getTopSales(int,int);
CREATE OR REPLACE FUNCTION getTopSales(year1 INTEGER, year2 INTEGER) RETURNS TABLE(Year numeric, Film
varchar, Sales numeric, Id INT) AS $$
BEGIN
    CREATE OR REPLACE VIEW filmsYear AS
        SELECT EXTRACT(year FROM orders.orderdate) as myYear, movies.movietitle as title, odt.quantity as
qtt, movies.movieid AS myID
        FROM products, orders, orderdetail AS odt, imdb_movies AS movies
        WHERE products.prod_id = odt.prod_id AND products.movieid = movies.movieid AND odt.orderid =
orders.orderid
    ;
    CREATE OR REPLACE VIEW sumYear AS
        SELECT myYear, title, sum(qtt) as freq, myID
        FROM filmsYear
        GROUP BY myYear, title, myID
    ;

    RETURN QUERY
    SELECT sumYear.myYear, sumYear.title, sumYear.freq, sumYear.myID
    FROM sumYear
    WHERE sumYear.myYear >= year1 AND sumYear.myYear <= year2
    ORDER BY sumYear.freq DESC;

END;
$$ LANGUAGE plpgsql;
```

2. En la página principal añadimos un botón para poder seleccionar los topSales de cada año (1 película por año limitado)



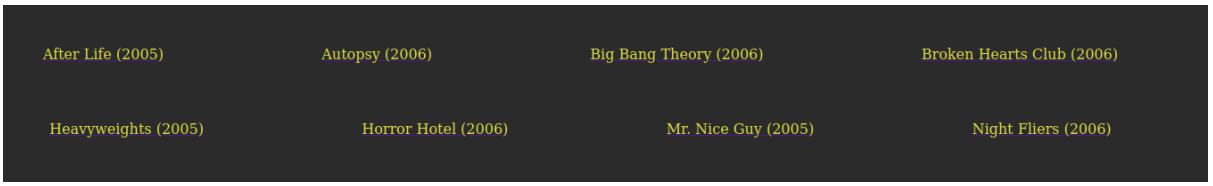
3. En la página principal añadimos unos botones para poder seleccionar manualmente dos años y que nuestra función nos devuelva las películas con más ventas entre los años seleccionados. Además se puede seleccionar la cantidad de películas a devolver por año (top 5=5 por año).

Elena Balseiro García
Paula Beltrán Marianini
Grupo: 1392
Pareja: 08

4. En database.py y vdv.py se hicieron los cambios necesarios para esta funcionalidad.

k) Se ha implementado correctamente el login y el register usando la base de datos.

l) Para las películas que se muestran en la pantalla principal sin ningún tipo de filtro seleccionamos de la base de datos las películas correspondientes a los años 2005 y 2006.



m)

n) En cuanto a las valoraciones. En los detalles de una película aparece la media y el recuento de votos de esa película. Si además el usuario está logueado, le aparecen dos opciones:

1. Valorar la película:
    a. Si no había valorado antes esa película entonces se añadirá su nueva valoración a la base de datos y saltará el trigger que actualizará la media y el recuento de votos.
    b. Si ya había valorado antes esa película entonces se actualizará su voto antiguo con el voto nuevo en la base de datos y saltará el trigger que actualizará la media y el recuento de votos.
2. Borrar su valoración: en el caso de que el usuario hubiera valorado esa película, se borrará su voto de la base de datos y saltará el trigger que actualizará la media y el recuento de votos.