Sakila Data Warehouse

Script de Sakila_SA:

```
use master;
go

create database sakila_SA;
go

use sakila_SA;
go

drop table actor;
go

drop table address;
go

drop table category;
go

drop table city;
go

drop table country;
go

drop table customer;
go

drop table film;
go

drop table film_actor;
go

drop table film_category;
go

drop table film_text;
go

drop table inventory;
go

drop table language;
go

drop table payment;
go

drop table rental;
go

drop table staff;
go
```

```
drop table store;
go

create table actor(
        actor_id smallint not null,
        first_name varchar(45) not null,
        last_name varchar(45) not null,
        last_update datetime not null
);
go

create table address(
        address_id smallint not null,
        address varchar(50) not null,
        address2 varchar(50) null,
        district varchar(20) not null,
        city_id smallint not null,
        postal_code varchar(10) null,
        phone varchar(20) not null,
        last_update datetime not null
);
go

create table category(
        category_id tinyint not null,
        name varchar(25) not null,
        last_update datetime not null
);
go

create table city(
        city_id smallint not null,
        city varchar(50) not null,
        country_id smallint not null,
        last_update datetime not null
);
go

create table country(
        country_id smallint not null,
        country varchar(50) not null,
        last_update datetime not null
);
go

create table customer(
        customer_id smallint not null,
        store_id tinyint not null,
        first_name varchar(45) not null,
        last_name varchar(45) not null,
        email varchar(50) null,
        address_id smallint not null,
        active bit not null,
        create_date datetime not null,
        last_update datetime not null
);
go
```

```
create table film(
        film_id smallint not null,
        title varchar(255) not null,
        description text null,
        release_year smallint null,
        language_id tinyint not null,
        original_language_id tinyint null,
        rental_duration tinyint not null,
        rental_rate decimal(4,2) not null,
        length smallint null,
        replacement_cost decimal(5,2) not null,
        rating varchar(6) not null,
        special_features varchar(100) null,
        last_update datetime not null,
);
go

create table film_actor(
        actor_id smallint not null,
        film_id smallint not null,
        last_update datetime not null
);
go

create table film_category(
        film_id smallint not null,
        category_id tinyint not null,
        last_update datetime not null
);
go

create table film_text(
        film_id smallint not null,
        title varchar(255) not null,
        description text
);
go

create table inventory(
        inventory_id smallint not null,
        film_id smallint not null,
        store_id tinyint not null,
        last_update datetime not null
);
go

create table language(
        language_id tinyint not null,
        name char(20) not null,
        last_update datetime not null
);
go

create table payment(
        payment_id smallint not null,
        customer_id smallint not null,
        staff_id tinyint not null,
        rental_id int null,
```

```
        amount decimal(5,2) not null,
        payment_date datetime not null,
        last_update datetime not null
);
go

create table rental(
        rental_id int not null,
        rental_date datetime not null,
        inventory_id smallint not null,
        customer_id smallint not null,
        return_date datetime null,
        staff_id tinyint not null,
        last_update datetime not null
);
go

create table staff(
        staff_id tinyint not null,
        first_name varchar(45) not null,
        last_name varchar(45) not null,
        address_id smallint not null,
        picture varbinary(max) null,
        email varchar(50) null,
        store_id tinyint not null,
        active bit not null,
        username varchar(16) not null,
        password varchar(40) null,
        last_update datetime not null
);
go

create table store(
        store_id tinyint not null,
        manager_staff_id tinyint not null,
        address_id smallint not null,
        last_update datetime not null
);
go
```

Script de Sakila_DW:

```
use master;
go

create database sakila_DW;
go

use sakila_DW;
go

create table dim_category(
        category_id tinyint identity(1,1) not null,
        name varchar(25) not null,
        last_update int not null
);
go
```

```
create table dim_city(
	city_id smallint identity(1,1) not null,
	city varchar(50) not null,
	country_id smallint not null,
	last_update int not null
);
go

create table dim_country(
	country_id smallint identity(1,1) not null,
	country varchar(50) not null,
	last_update int not null
);
go

create table dim_customer(
	customer_id smallint identity(1,1) not null,
	first_name varchar(45) not null,
	last_name varchar(45) not null,
	phone varchar(20) not null,
	email varchar(50) null,
	district_id smallint not null,
	active bit not null,
	create_date int not null,
	last_update int not null
);
go

create table dim_district(
	district_id smallint identity(1,1) not null,
	district varchar(20) not null,
	city_id smallint not null,
	last_update int not null
);
go

create table dim_film(
	film_id smallint identity(1,1) not null,
	title varchar(255) not null,
	description text null,
	release_year smallint null,
	rental_duration tinyint not null,
	rental_rate decimal(4,2) not null,
	length smallint null,
	replacement_cost decimal(5,2) not null,
	rating varchar(6) not null,
	special_features varchar(100) null,
	last_update int not null,
);
go

create table dim_film_category(
	film_id smallint not null,
	category_id tinyint not null,
	last_update int not null
);
go
```

```
create table dim_payment(
        payment_id smallint identity(1,1) not null,
        customer_id smallint not null,
        rental_id int null,
        amount decimal(5,2) not null,
        payment_date int not null,
        last_update int not null
);
go

create table dim_rental(
        rental_id int identity(1,1) not null,
        rental_date int not null,
        film_id smallint not null,
        customer_id smallint not null,
        return_date int null,
        last_update int not null
);
go

create table dim_time(
        id int identity(1,1) not null,
        registerdate datetime not null,
        year smallint not null,
        month varchar(12) not null,
        day smallint not null,
        dayname varchar(12) not null,
        quarter varchar(12) not null
);
go

alter table dim_category add constraint pk_category primary key(category_id);
alter table dim_city add constraint pk_city primary key(city_id);
alter table dim_country add constraint pk_country primary key(country_id);
alter table dim_customer add constraint pk_customer primary key(customer_id);
alter table dim_district add constraint pk_district primary key(district_id);
alter table dim_film add constraint pk_film primary key(film_id);
alter table dim_payment add constraint pk_payment primary key(payment_id);
alter table dim_rental add constraint pk_rental primary key(rental_id);
alter table dim_time add constraint pk_time primary key(id);

alter table dim_city add constraint fk_city_country foreign key(country_id) references
dim_country(country_id);
alter table dim_customer add constraint fk_cust_district foreign key(district_id) references
dim_district(district_id);
alter table dim_district add constraint fk_dist_city foreign key(city_id) references
dim_city(city_id);
alter table dim_film_category add constraint fk_cat_category foreign key(category_id)
references dim_category(category_id);
alter table dim_film_category add constraint fk_cat_film foreign key(film_id) references
dim_film(film_id);
alter table dim_payment add constraint fk_pay_customer foreign key(customer_id)
references dim_customer(customer_id);
alter table dim_payment add constraint fk_pay_rental foreign key(rental_id) references
dim_rental(rental_id);
alter table dim_rental add constraint fk_rent_customer foreign key(customer_id)
references dim_customer(customer_id);
```

```
alter table dim_rental add constraint fk_rent_film foreign key(film_id) references
dim_film(film_id);

alter table dim_customer add constraint fk_cust_time foreign key(create_date) references
dim_time(id);
alter table dim_payment add constraint fk_pay_time foreign key(payment_date)
references dim_time(id);
alter table dim_rental add constraint fk_rent_time_rental foreign key(rental_date)
references dim_time(id);
alter table dim_rental add constraint fk_rent_time_return foreign key(return_date)
references dim_time(id);

alter table dim_category add constraint fk_cat_last_update foreign key(last_update)
references dim_time(id);
alter table dim_city add constraint fk_city_last_update foreign key(last_update) references
dim_time(id);
alter table dim_country add constraint fk_country_last_update foreign key(last_update)
references dim_time(id);
alter table dim_customer add constraint fk_cust_last_update foreign key(last_update)
references dim_time(id);
alter table dim_district add constraint fk_dist_last_update foreign key(last_update)
references dim_time(id);
alter table dim_film add constraint fk_film_last_update foreign key(last_update)
references dim_time(id);
alter table dim_film_category add constraint fk_film_cat_last_update foreign
key(last_update) references dim_time(id);
alter table dim_payment add constraint fk_pay_last_update foreign key(last_update)
references dim_time(id);
alter table dim_rental add constraint fk_rent_last_update foreign key(last_update)
references dim_time(id);
```

Script de proceso ETL

```
-- Extraer fecha de registro del cliente (S1).
select distinct(cust.create_date),
      cast(year(cust.create_date) as smallint ) as year,
      case MONTH(cust.create_date)
                  when 1 then 'January'   when 2 then 'February'   when 3 then 'March'
                  when 4 then 'April'     when 5 then 'May'        when 6 then 'June'
                  when 7 then 'July'      when 8 then 'August'     when 9 then 'September'
                  when 10 then 'October'  when 11 then 'November'  when 12 then
'December'
            end as month,
            cast(day(cust.create_date) as smallint ) as day,
            case datepart(WEEKDAY, cust.create_date)
              when 1 then 'Sunday'    when 2 then 'Monday'
                  when 3 then 'Tuesday'   when 4 then 'Wednesday'
                  when 5 then 'Thursday'  when 6 then 'Friday'
                  when 7 then 'Saturday'
            end as dayname,
            case
                  when MONTH(cust.create_date) <= 3 then 'Quarter 1'
                  when MONTH(cust.create_date) <= 6 then 'Quarter 2'
                  when MONTH(cust.create_date) <= 9 then 'Quarter 3'
                  when MONTH(cust.create_date)  <= 12 then 'Quarter 4'
            end as quarter
from sakila_SA.dbo.customer cust
```

```
where cust.create_date not in (select registerdate from sakila_DW.dbo.dim_time);

-- Extraer fecha de pago (S2).
select distinct(pay.payment_date),
      cast(year(pay.payment_date) as smallint ) as year,
      case MONTH(pay.payment_date)
                  when 1 then 'January'   when 2 then 'February'   when 3 then 'March'
                  when 4 then 'April'     when 5 then 'May'        when 6 then 'June'
                  when 7 then 'July'      when 8 then 'August'     when 9 then 'September'
                  when 10 then 'October'  when 11 then 'November'  when 12 then
'December'
            end as month,
            cast(day(pay.payment_date) as smallint ) as day,
            case datepart(WEEKDAY, pay.payment_date)
              when 1 then 'Sunday'    when 2 then 'Monday'
                  when 3 then 'Tuesday'   when 4 then 'Wednesday'
                  when 5 then 'Thursday'  when 6 then 'Friday'
                  when 7 then 'Saturday'
            end as dayname,
            case
                  when MONTH(pay.payment_date) <= 3 then 'Quarter 1'
                  when MONTH(pay.payment_date) <= 6 then 'Quarter 2'
                  when MONTH(pay.payment_date) <= 9 then 'Quarter 3'
                  when MONTH(pay.payment_date)  <= 12 then 'Quarter 4'
            end as quarter
from sakila_SA.dbo.payment pay
where pay.payment_date not in (select registerdate from sakila_DW.dbo.dim_time);

-- Extraer fecha de arrendamiento (S3).
select distinct(rent.rental_date),
      cast(year(rent.rental_date) as smallint ) as year,
      case MONTH(rent.rental_date)
                  when 1 then 'January'   when 2 then 'February'   when 3 then 'March'
                  when 4 then 'April'     when 5 then 'May'        when 6 then 'June'
                  when 7 then 'July'      when 8 then 'August'     when 9 then 'September'
                  when 10 then 'October'  when 11 then 'November'  when 12 then
'December'
            end as month,
            cast(day(rent.rental_date) as smallint ) as day,
            case datepart(WEEKDAY, rent.rental_date)
              when 1 then 'Sunday'    when 2 then 'Monday'
                  when 3 then 'Tuesday'   when 4 then 'Wednesday'
                  when 5 then 'Thursday'  when 6 then 'Friday'
                  when 7 then 'Saturday'
            end as dayname,
            case
                  when MONTH(rent.rental_date) <= 3 then 'Quarter 1'
                  when MONTH(rent.rental_date) <= 6 then 'Quarter 2'
                  when MONTH(rent.rental_date) <= 9 then 'Quarter 3'
                  when MONTH(rent.rental_date)  <= 12 then 'Quarter 4'
            end as quarter
from sakila_SA.dbo.rental rent
where rent.rental_date not in (select registerdate from sakila_DW.dbo.dim_time);

-- Extraer fecha de devolución (S4).
select distinct(rent.return_date),
      cast(year(rent.return_date) as smallint ) as year,
      case MONTH(rent.return_date)
```

```
                when 1 then 'January'   when 2 then 'February'   when 3 then 'March'
                when 4 then 'April'     when 5 then 'May'        when 6 then 'June'
                when 7 then 'July'      when 8 then 'August'     when 9 then 'September'
                when 10 then 'October'  when 11 then 'November'  when 12 then
'December'
        end as month,
        cast(day(rent.return_date) as smallint ) as day,
        case datepart(WEEKDAY, rent.return_date)
          when 1 then 'Sunday'    when 2 then 'Monday'
                when 3 then 'Tuesday'   when 4 then 'Wednesday'
                when 5 then 'Thursday'  when 6 then 'Friday'
                when 7 then 'Saturday'
        end as dayname,
        case
                when MONTH(rent.return_date) <= 3 then 'Quarter 1'
                when MONTH(rent.return_date) <= 6 then 'Quarter 2'
                when MONTH(rent.return_date) <= 9 then 'Quarter 3'
                when MONTH(rent.return_date)  <= 12 then 'Quarter 4'
        end as quarter
from sakila_SA.dbo.rental rent
where rent.return_date not in (select registerdate from sakila_DW.dbo.dim_time);


-- Extraer la última fecha de modificación para las columnas (cambiar el nombre de la
columna) (S 5-13).
select distinct(time.last_update),
     cast(year(time.last_update) as smallint ) as year,
     case MONTH(time.last_update)
                when 1 then 'January'   when 2 then 'February'   when 3 then 'March'
                when 4 then 'April'     when 5 then 'May'        when 6 then 'June'
                when 7 then 'July'      when 8 then 'August'     when 9 then 'September'
                when 10 then 'October'  when 11 then 'November'  when 12 then
'December'
        end as month,
        cast(day(time.last_update) as smallint ) as day,
        case datepart(WEEKDAY, time.last_update)
          when 1 then 'Sunday'    when 2 then 'Monday'
                when 3 then 'Tuesday'   when 4 then 'Wednesday'
                when 5 then 'Thursday'  when 6 then 'Friday'
                when 7 then 'Saturday'
        end as dayname,
        case
                when MONTH(time.last_update) <= 3 then 'Quarter 1'
                when MONTH(time.last_update) <= 6 then 'Quarter 2'
                when MONTH(time.last_update) <= 9 then 'Quarter 3'
                when MONTH(time.last_update)  <= 12 then 'Quarter 4'
        end as quarter
from sakila_SA.dbo.[column] time
where time.last_update not in (select registerdate from sakila_DW.dbo.dim_time);

-- Extraer datos de categorías (S 14).
select
        cat.name,
        (
                select
                        time.id
                from
                        sakila_DW.dbo.dim_time time
```

```
                where
                        time.registerdate = cat.last_update
        ) as last_update_id
from
        sakila_SA.dbo.category cat;

-- Extraer datos de países (S 14).
select
        cou.country,
        (
                select
                        time.id
                from
                        sakila_DW.dbo.dim_time time
                where
                        time.registerdate = cou.last_update
        ) as last_update_id
from
        sakila_SA.dbo.country cou;

-- Extraer datos de filmes (S 14).
select
        film.title,
        film.description,
        film.release_year,
        film.rental_duration,
        film.rental_rate,
        film.length,
        film.replacement_cost,
        film.rating,
        film.special_features,
        (
                select
                        time.id
                from
                        sakila_DW.dbo.dim_time time
                where
                        time.registerdate = film.last_update
        ) as last_update_id
from
        sakila_SA.dbo.film film;

-- Extraer datos de ciudades (S 15).
select
        cit.city,
        cit.country_id,
        (
                select
                        time.id
                from
                        sakila_DW.dbo.dim_time time
                where
                        time.registerdate = cit.last_update
        ) as last_update_id
from
        sakila_SA.dbo.city cit;

-- Extraer IDs de filmes y categorías (S 15).
```

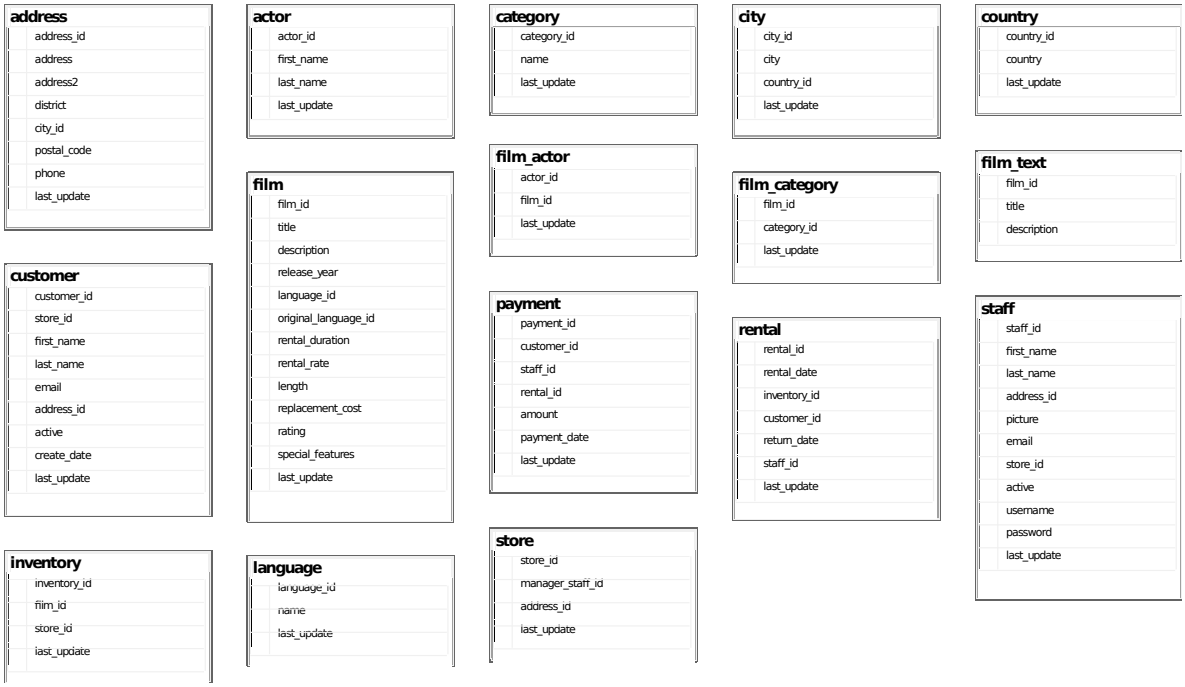
```sql
select
	ficat.film_id,
	ficat.category_id,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = ficat.last_update
	) as last_update_id
from
	sakila_SA.dbo.film_category ficat;

-- Extraer datos de distrito desde la dirección (S 16).
select
	addr.district,
	addr.city_id,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = addr.last_update
	) as last_update_id
from
	sakila_SA.dbo.address addr;

-- Extraer datos de cliente (S 17).
select
	cust.first_name,
	cust.last_name,
	addr.phone,
	cust.email,
	cust.address_id,
	cust.active,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = cust.create_date
	) as create_date_id,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = cust.last_update
	) as last_update_id
from
	sakila_SA.dbo.customer cust,
	sakila_SA.dbo.address addr
where
	cust.address_id = addr.address_id;
```

```sql
-- Extraer datos de arrendamientos (S 18).
select
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = rent.rental_date
	) as rental_date_id,
	film.film_id,
	rent.customer_id,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = rent.return_date
	) as return_date_id,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = rent.last_update
	) as last_update_id
from
	sakila_SA.dbo.film film,
	sakila_SA.dbo.inventory inv,
	sakila_SA.dbo.rental rent
where
	inv.film_id = film.film_id
	and
	inv.inventory_id = rent.inventory_id;

-- Extraer datos de pagos (S 19).
select
	pay.customer_id,
	pay.rental_id,
	pay.amount,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = pay.payment_date
	) as payment_date_id,
	(
		select
			time.id
		from
			sakila_DW.dbo.dim_time time
		where
			time.registerdate = pay.last_update
```

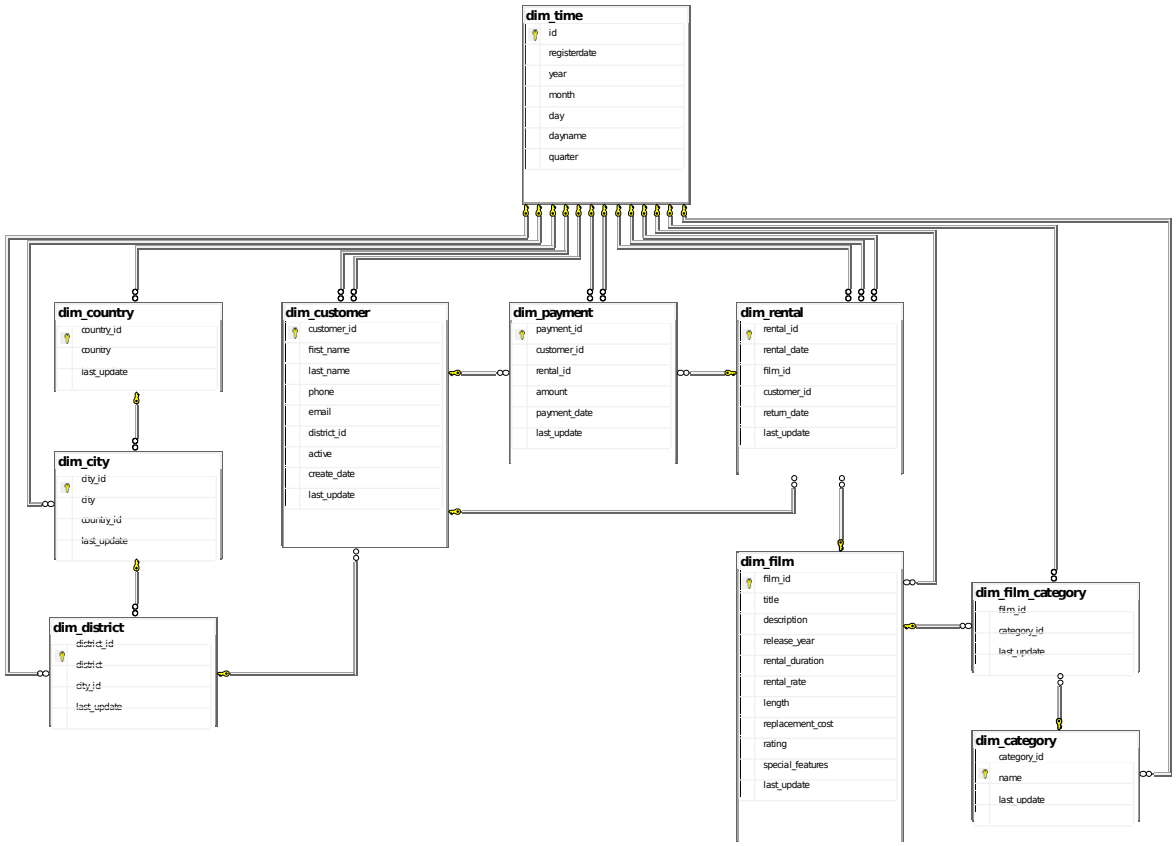
```
        ) as last_update_id
from
        sakila_SA.dbo.payment pay;

-- Limpiar starting area.
delete from actor;
go

delete from address;
go

delete from category;
go

delete from city;
go

delete from country;
go

delete from customer;
go

delete from film;
go

delete from film_actor;
go

delete from film_category;
go

delete from film_text;
go

delete from inventory;
go

delete from language;
go

delete from payment;
go

delete from rental;
go

delete from staff;
go

delete from store;
go
```

Diagramas de BD:

- Sakila_SA:



- Sakila_DW:



Scripts de consultas y justificaciones:

1. Se puede determinar el margen de ingresos que tiene la empresa en varias dimensiones, por rango de tiempo, por zonas, por genero de las películas que se arrendan.

```
-- Por rango de tiempo.
select
        sum(pay.amount) as total
from
        dim_payment pay,
        dim_rental rent,
        dim_time time
where
        (pay.rental_id = rent.rental_id)
        and
        (time.id = rent.rental_date)
        and
        (time.registerdate > '2005-05-24')
        and
        (time.registerdate < '2006-02-15');

-- Por zonas (país, ciudad y distrito).
select
        country.country,
        city.city,
        dist.district,
        sum(pay.amount) as total
from
        dim_payment pay,
        dim_customer cust,
        dim_district dist,
        dim_city city,
        dim_country country
where
        (country.country_id = city.country_id)
        and
        (dist.city_id = country.country_id)
        and
        (dist.district_id = cust.district_id)
        and
        (pay.customer_id = cust.customer_id)
group by
        country.country,
        city.city,
        dist.district
order by
        country.country,
        city.city,
        dist.district;

-- Por género de las películas (categorías).
select
        cat.name as category,
        sum(pay.amount) as total
from
        dim_payment pay,
        dim_rental rent,
        dim_film film,
        dim_film_category ficat,
        dim_category cat
```

```
where
        (cat.category_id = ficat.category_id)
        and
        (film.film_id = ficat.film_id)
        and
        (film.film_id = rent.film_id)
        and
        (pay.rental_id = rent.rental_id)
group by
        cat.name;
```

Justificación: Las consultas anteriores extraen los valores de la cantidad total de ingresos, es decir, la cantidad total de lo que pagaron sus clientes por los filmes, ya sea por un rango de tiempo (especificar las fechas que encierran el rango), por área (país, ciudad o distrito donde viven sus clientes, se puede especificar), o por el género de la película (la categoría que está asociada).

2. Por tiempo determinar que periodos del año generan mayor movimiento de operaciones y si estas coinciden con fechas calendarios específicas como fiestas patrias, torneos deportivos, vacaciones escolares.

```
select
        time.registerdate as period,
        count(rent.rental_id) as rental_amount
from
        dim_rental rent,
        dim_time time
where
        (time.id = rent.rental_date)
        and
        (time.registerdate > '2005-05-24')
        and
        (time.registerdate < '2006-02-15')
group by
        time.registerdate
order by
        count(rent.rental_id) desc;
```

Justificación: La consulta anterior extrae la cantidad total de arrendamientos de las películas que representa los movimientos de las operaciones, y se debe especificar el rango de fecha para determinar un periodo el cual se interesa revisar. Se ordena según la cantidad total de arrendamientos de mayor a menor.

3. Determinar los géneros de filmes más buscados y cuales los menos, como para determinar correcciones en los procesos de adquisición de nuevo material, o en su defecto eliminar material que no registra movimientos, recuerde considerar la dimTiempo.

```
select
        cat.name as category,
        count(rent.rental_id) as rental_amount
from
        dim_rental rent,
```

```
        dim_film film,
        dim_film_category ficat,
        dim_category cat
where
        (cat.category_id = ficat.category_id)
        and
        (film.film_id = ficat.film_id)
        and
        (film.film_id = rent.film_id)
group by
        cat.name
order by
        count(rent.rental_id) desc;
```

Justificación: La consulta anterior extrae la cantidad total de arrendamientos de las películas por cada categoría, que representa la cantidad de búsquedas que queremos conocer. Se ordena según la cantidad total de arrendamientos de mayor a menor.