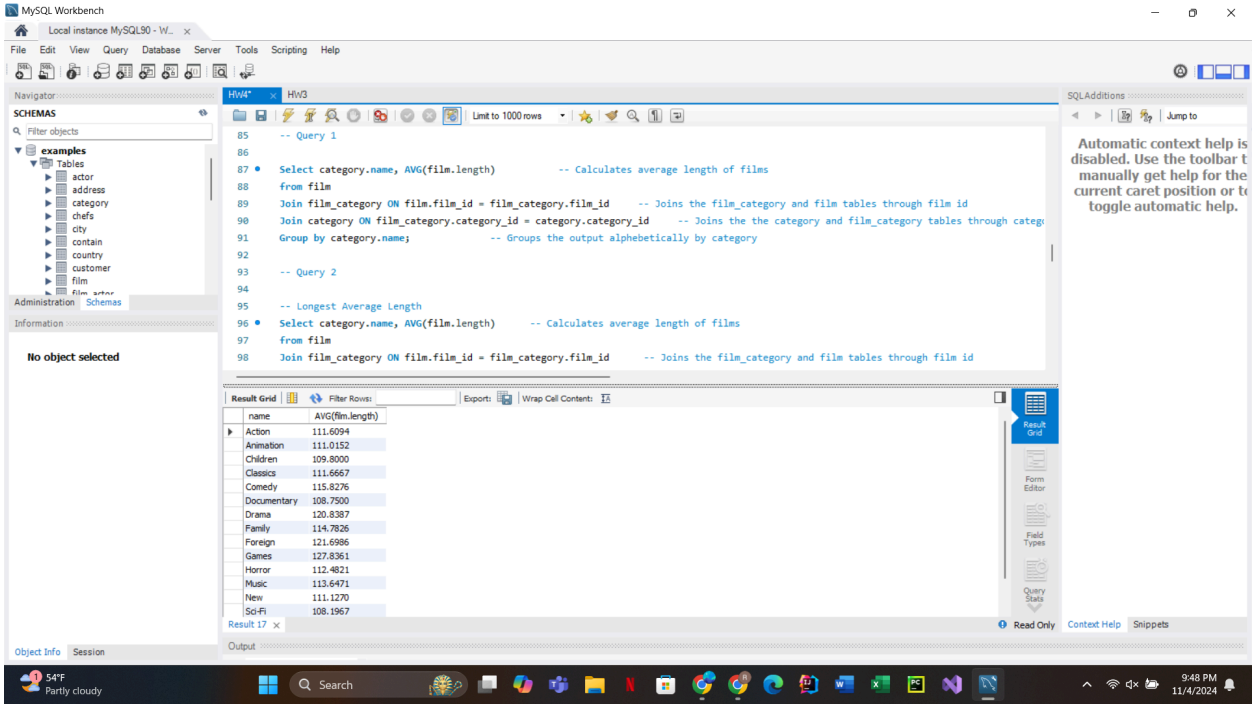
Title: DB Assignment 4

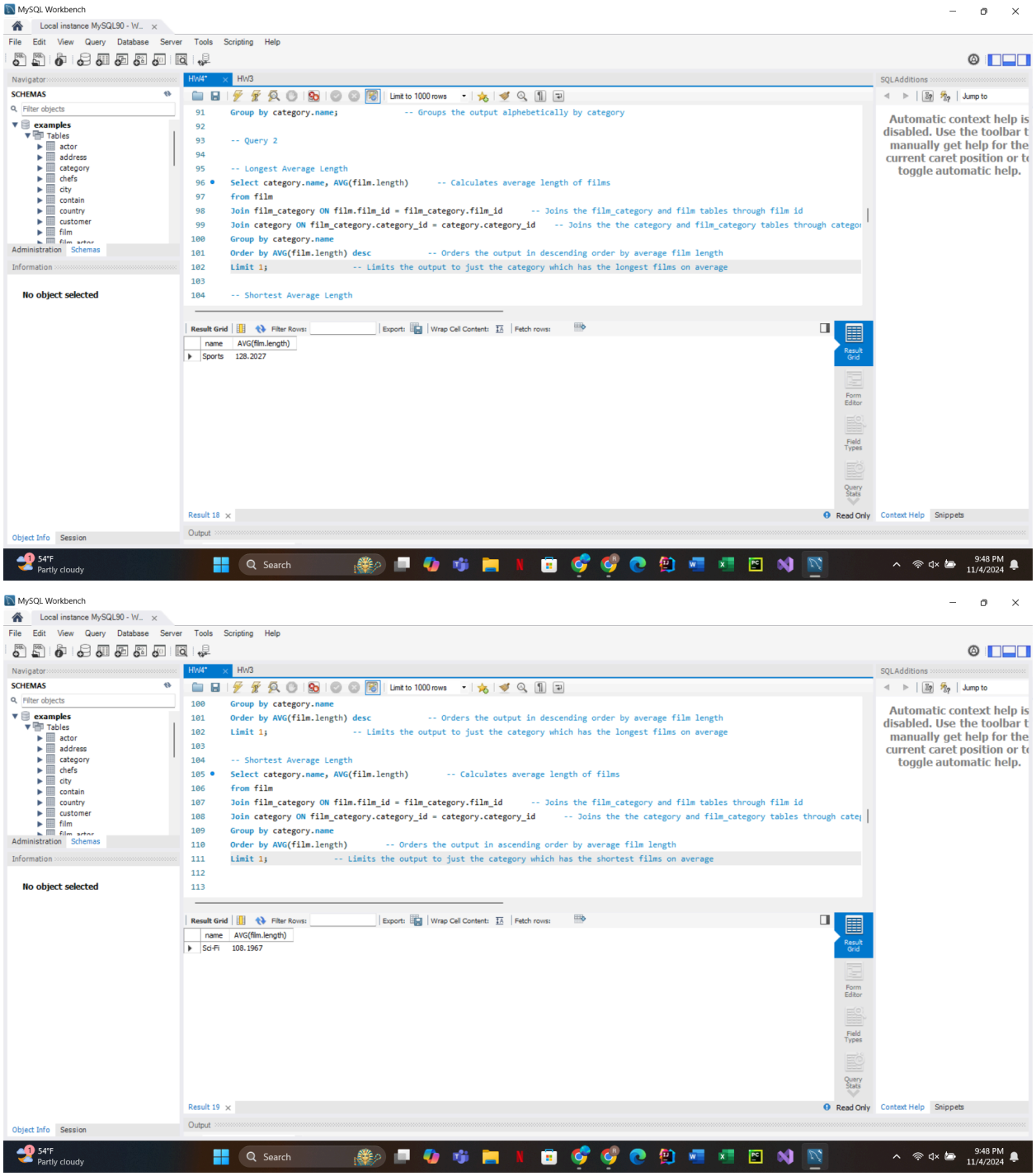
Your Name: Ryan Smith

Date: 11/4/2024

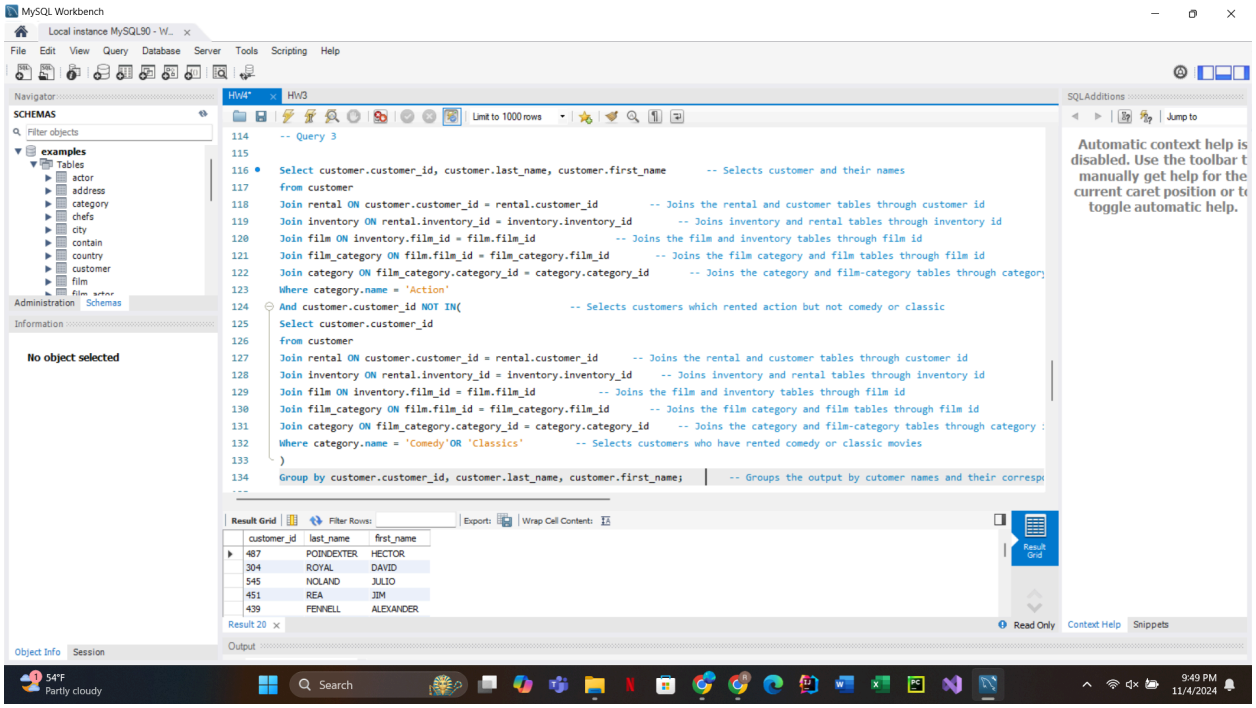
Github link: https://github.com/rsmith1388/Databases.git



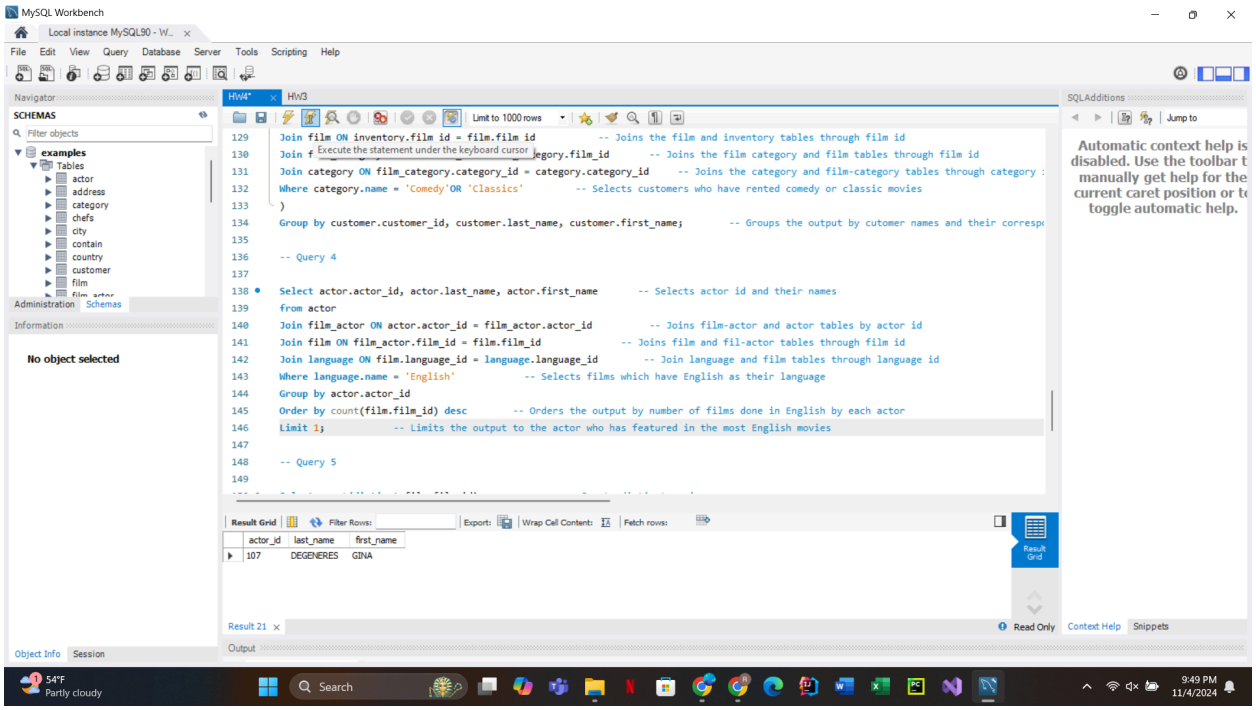
Query 1: The query calculates the average length of films and orders the output in alphabetical order based on category name.



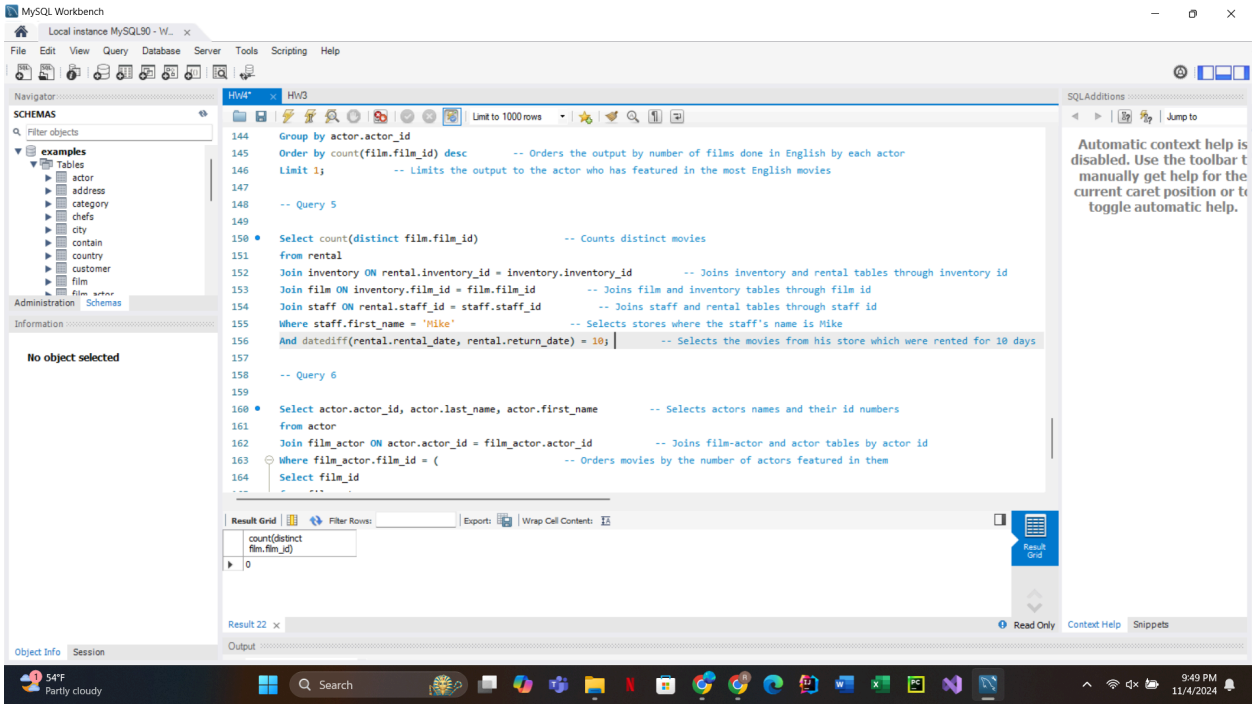
Query 2: The two queries calculate the films which have the highest and lowest length times on average through table joins.



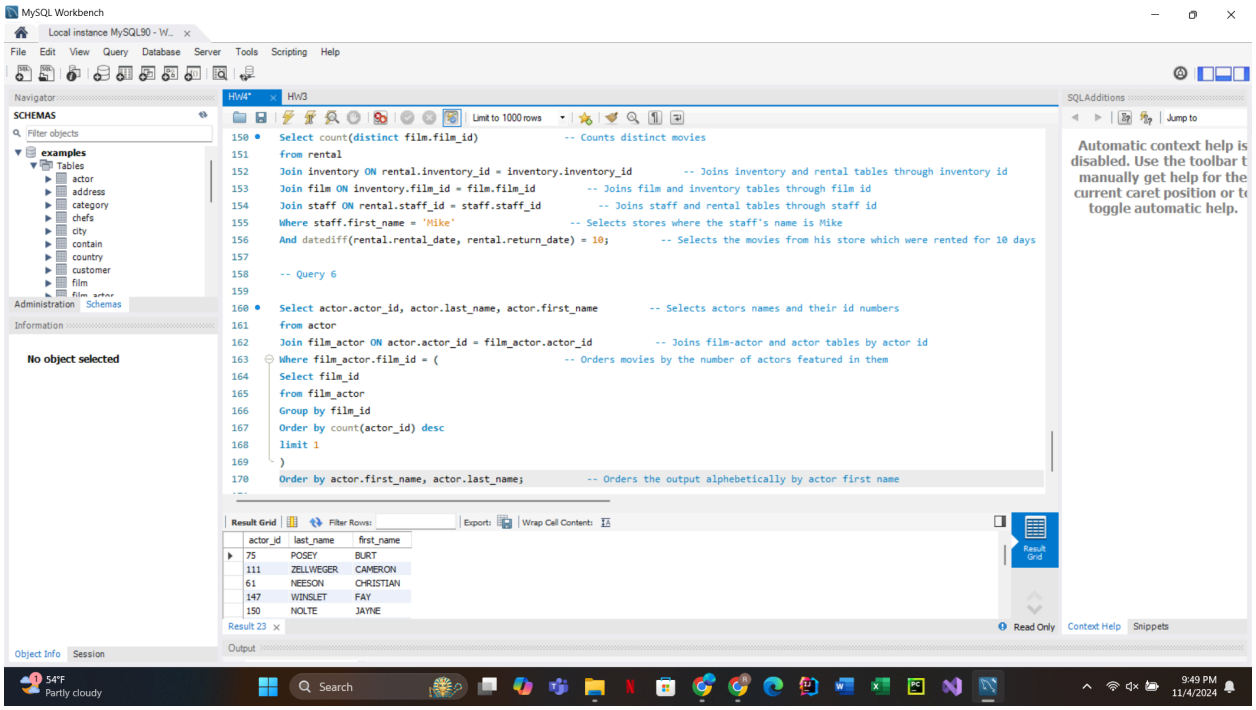
Query 3: The query uses table joins to identify customers who have rented action movies but not comedy or classic movies.



Query 4: The query identifies actors who have appeared in the most English movies and returns the one who has appeared the most.



Query 5: The query counts distinct movies and calculates how many of them were rented for 10 days from the store where Mike works.



Query 6: The query lists movies based on cast size and returns the actor who has appeared in the most movies with the largest casts.

```
Alter table actor add primary key (actor_id);
Alter table address add primary key (address_id);
Alter table category add primary key (category_id);
Alter table city add primary key (city_id);
Alter table country add primary key (country_id);
Alter table customer add primary key (customer_id);
Alter table film add primary key (film_id);
Alter table film_actor add primary key (actor_id, film_id);
Alter table film_category add primary key (film_id, category_id);
Alter table inventory add primary key (inventory_id);
Alter table language add primary key (language_id);
Alter table payment add primary key (payment_id);
Alter table rental add primary key (rental_id);
Alter table staff add primary key (staff_id);
Alter table store add primary key (store_id);
-- Primary Keys
Alter table address
Add constraint address_fk foreign key (city_id) references city(city_id);
Alter table city
Add constraint city_fk foreign key (country_id) references country(country_id);
Alter table customer
Add constraint customer_add foreign key (address_id) references address(address_id);
Alter table customer
Add constraint customer_store foreign key (store_id) references store(store_id);
Alter table film
Add constraint film_fk foreign key (language_id) references language(language_id);
Alter table film_actor
Add constraint actorID foreign key (actor_id) references actor(actor_id);
Alter table film_actor
Add constraint actorFilmID foreign key (film_id) references film(film_id);
Alter table rental
Add constraint rentalStaff foreign key (staff_id) references staff(staff_id);
Alter table rental
Add constraint rentalCustomer foreign key (customer_id) references customer(customer_id);
Alter table rental
Add constraint rentalInv foreign key (inventory_id) references inventory(inventory_id);
Alter table staff
Add constraint staffAdd foreign key (address_id) references address(address_id);
Alter table staff
Add constraint staffStore foreign key (store_id) references store(store_id);
Alter table store
Add constraint storeAdd foreign key (address_id) references address(address_id);
Alter table film_category
Add constraint categoryFilm foreign key (film_id) references film(film_id);
```

```
Alter table film_category
Add constraint categoryID foreign key (category_id) references category(category_id);
Alter table inventory
Add constraint inventoryFilm foreign key (film_id) references film(film_id);
Alter table inventory
Add constraint inventoryStore foreign key (store_id) references store(store_id);
Alter table payment
Add constraint paymentCustomer foreign key (customer_id) references customer(customer_id);
Alter table payment
Add constraint paymentStaff foreign key (staff_id) references staff(staff_id);
Alter table payment
Add constraint paymentRental foreign key (rental_id) references rental(rental_id);
-- Foreign Keys
Alter table category
Add constraint category_type Check(name IN('Animation', 'Comedy', 'Family', 'Foreign', 'Sci-Fi',
'Travel', 'Children', 'Drama', 'Horror', 'Action', 'Classics', 'Games', 'New', 'Documentary', 'Sports',
'Music'));
Alter table film
Add constraint film_special Check(special_features IN('Behind the Scenes',
'Commentaries','Deleted Scenes', 'Trailers'));
Alter Table rental
Add constraint dates Check(rental_date >= '2004-01-01');
Alter Table rental
Add constraint returnDates Check(return_date >= '2004-01-01');
Alter Table staff
Add constraint activeStaff Check(active IN(0,1));
Alter Table customer
Add constraint activeCustomer Check(active IN(0,1));
Alter table film
Add constraint rentalDuration Check(rental_duration between 2 and 8);
Alter table film
Add constraint rentalRate Check(rental_rate between 0.99 and 6.99);
Alter table film
Add constraint filmLength Check(length between 30 and 200);
Alter table film
Add constraint film_rating Check(rating IN('PG', 'G','NC-17', 'PG-13','R'));
Alter table film
Add constraint replacement Check(replacement_cost between 5.00 and 100.00);
Alter table payment
Add constraint paymentAmount Check(amount >= 0);
-- Constraints


-- Query 1
```

```
Select category.name, AVG(film.length)          -- Calculates average length of films
from film
Join film_category ON film.film_id = film_category.film_id     -- Joins the film_category and film tables through film id
Join category ON film_category.category_id = category.category_id     -- Joins the the category and film_category tables through category id
Group by category.name;              -- Groups the output alphebetically by category

-- Query 2

-- Longest Average Length
Select category.name, AVG(film.length)      -- Calculates average length of films
from film
Join film_category ON film.film_id = film_category.film_id      -- Joins the film_category and film tables through film id
Join category ON film_category.category_id = category.category_id    -- Joins the the category and film_category tables through category id
Group by category.name
Order by AVG(film.length) desc           -- Orders the output in descending order by average film length
Limit 1;                -- Limits the output to just the category which has the longest films on average

-- Shortest Average Length
Select category.name, AVG(film.length)        -- Calculates average length of films
from film
Join film_category ON film.film_id = film_category.film_id      -- Joins the film_category and film tables through film id
Join category ON film_category.category_id = category.category_id      -- Joins the the category and film_category tables through category id
Group by category.name
Order by AVG(film.length)        -- Orders the output in ascending order by average film length
Limit 1;             -- Limits the output to just the category which has the shortest films on average


-- Query 3

Select customer.customer_id, customer.last_name, customer.first_name       -- Selects customer and their names
from customer
Join rental ON customer.customer_id = rental.customer_id         -- Joins the rental and customer tables through customer id
```

```
Join inventory ON rental.inventory_id = inventory.inventory_id        -- Joins inventory and rental tables through inventory id
Join film ON inventory.film_id = film.film_id              -- Joins the film and inventory tables through film id
Join film_category ON film.film_id = film_category.film_id        -- Joins the film category and film tables through film id
Join category ON film_category.category_id = category.category_id       -- Joins the category and film-category tables through category id
Where category.name = 'Action'
And customer.customer_id NOT IN(                  -- Selects customers which rented action but not comedy or classic
Select customer.customer_id
from customer
Join rental ON customer.customer_id = rental.customer_id      -- Joins the rental and customer tables through customer id
Join inventory ON rental.inventory_id = inventory.inventory_id     -- Joins inventory and rental tables through inventory id
Join film ON inventory.film_id = film.film_id           -- Joins the film and inventory tables through film id
Join film_category ON film.film_id = film_category.film_id       -- Joins the film category and film tables through film id
Join category ON film_category.category_id = category.category_id     -- Joins the category and film-category tables through category id
Where category.name = 'Comedy'OR 'Classics'         -- Selects customers who have rented comedy or classic movies
)
Group by customer.customer_id, customer.last_name, customer.first_name;        -- Groups the output by cutomer names and their corresponding id number

-- Query 4

Select actor.actor_id, actor.last_name, actor.first_name       -- Selects actor id and their names from actor
Join film_actor ON actor.actor_id = film_actor.actor_id          -- Joins film-actor and actor tables by actor id
Join film ON film_actor.film_id = film.film_id              -- Joins film and fil-actor tables through film id
Join language ON film.language_id = language.language_id        -- Join language and film tables through language id
Where language.name = 'English'           -- Selects films which have English as their language
Group by actor.actor_id
Order by count(film.film_id) desc        -- Orders the output by number of films done in English by each actor
Limit 1;           -- Limits the output to the actor who has featured in the most English movies
```

```
-- Query 5

Select count(distinct film.film_id)              -- Counts distinct movies
from rental
Join inventory ON rental.inventory_id = inventory.inventory_id        -- Joins inventory and rental
tables through inventory id
Join film ON inventory.film_id = film.film_id        -- Joins film and inventory tables through film id
Join staff ON rental.staff_id = staff.staff_id        -- Joins staff and rental tables through staff id
Where staff.first_name = 'Mike'                  -- Selects stores where the staff's name is Mike
And datediff(rental.rental_date, rental.return_date) = 10;        -- Selects the movies from his
store which were rented for 10 days

-- Query 6

Select actor.actor_id, actor.last_name, actor.first_name        -- Selects actors names and their id
numbers
from actor
Join film_actor ON actor.actor_id = film_actor.actor_id          -- Joins film-actor and actor tables
by actor id
Where film_actor.film_id = (                    -- Orders movies by the number of actors featured in
them
Select film_id
from film_actor
Group by film_id
Order by count(actor_id) desc
limit 1
)
Order by actor.first_name, actor.last_name;          -- Orders the output alphebetically by actor
first name
```