

Title: DB Assignment 4

Your Name: Ryan Smith

Date: 11/4/2024

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'examples' expanded, showing tables like actor, address, category, chefs, city, contain, country, customer, film, and film_actor. The main editor window contains two SQL queries. Query 1 calculates the average length of films by category, ordered alphabetically by category name. Query 2 calculates the longest average length of films. The 'Result Grid' at the bottom shows the output of Query 1, listing film categories and their average lengths.

```
-- Query 1
85
86
87 • Select category.name, AVG(film.length) -- Calculates average length of films
88 from film
89 Join film_category ON film.film_id = film_category.film_id -- Joins the film_category and film tables through film id
90 Join category ON film_category.category_id = category.category_id -- Joins the the category and film_category tables through category id
91 Group by category.name; -- Groups the output alphabetically by category
92
93
94
-- Query 2
95
96 • -- Longest Average Length
97 • Select category.name, AVG(film.length) -- Calculates average length of films
98 from film
99 Join film_category ON film.film_id = film_category.film_id -- Joins the film_category and film tables through film id
```

name	AVG(film.length)
Action	111.6094
Animation	111.0152
Children	109.8000
Classics	111.6667
Comedy	115.8276
Documentary	108.7500
Drama	120.8387
Family	114.7826
Foreign	121.6986
Games	127.8361
Horror	112.4821
Music	113.6471
New	111.1270
Sci-Fi	108.1967

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

MySQL Workbench

Local instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

examples

- Tables
 - actor
 - address
 - category
 - chefs
 - city
 - contain
 - country
 - customer
 - film
 - film_actor

Administration Schemas

Information

No object selected

SQL Editor

```

91 Group by category.name; -- Groups the output alphabetically by category
92
93 -- Query 2
94
95 -- Longest Average Length
96 Select category.name, AVG(film.length) -- Calculates average length of films
97 from film
98 Join film_category ON film.film_id = film_category.film_id -- Joins the film_category and film tables through film id
99 Join category ON film_category.category_id = category.category_id -- Joins the the category and film_category tables through category_id
100 Group by category.name
101 Order by AVG(film.length) desc -- Orders the output in descending order by average film length
102 Limit 1; -- Limits the output to just the category which has the longest films on average
103
104 -- Shortest Average Length
  
```

Result Grid

name	AVG(film.length)
Sports	128.2027

Result 18 x

Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Read Only Context Help Snippets

Object Info Session

54°F Partly cloudy

Search

9:48 PM 11/4/2024

MySQL Workbench

Local instance MySQL90 - W...

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

examples

- Tables
 - actor
 - address
 - category
 - chefs
 - city
 - contain
 - country
 - customer
 - film
 - film_actor

Administration Schemas

Information

No object selected

SQL Editor

```

100 Group by category.name
101 Order by AVG(film.length) desc -- Orders the output in descending order by average film length
102 Limit 1; -- Limits the output to just the category which has the longest films on average
103
104 -- Shortest Average Length
105 Select category.name, AVG(film.length) -- Calculates average length of films
106 from film
107 Join film_category ON film.film_id = film_category.film_id -- Joins the film_category and film tables through film id
108 Join category ON film_category.category_id = category.category_id -- Joins the the category and film_category tables through category_id
109 Group by category.name
110 Order by AVG(film.length) -- Orders the output in ascending order by average film length
111 Limit 1; -- Limits the output to just the category which has the shortest films on average
112
113
  
```

Result Grid

name	AVG(film.length)
So-Fi	108.1967

Result 19 x

Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Read Only Context Help Snippets

Object Info Session

54°F Partly cloudy

Search

9:48 PM 11/4/2024

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

MySQL Workbench interface showing Query 3. The query uses table joins to identify customers who have rented action movies but not comedy or classic movies. The result grid displays the following data:

customer_id	last_name	first_name
487	POINDEXTER	HECTOR
304	ROYAL	DAVID
545	NOLAND	JULIO
451	REA	JIM
439	FENNEL	ALEXANDER

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

MySQL Workbench interface showing Query 4. The query identifies actors who have appeared in the most English movies and returns the one who has appeared the most. The result grid displays the following data:

actor_id	last_name	first_name
107	DEGENERES	GINA

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

MySQL Workbench interface showing Query 5. The query is as follows:

```

144 Group by actor.actor_id
145 Order by count(film.film_id) desc -- Orders the output by number of films done in English by each actor
146 Limit 1; -- Limits the output to the actor who has featured in the most English movies
147
148 -- Query 5
149
150 • Select count(distinct film.film_id) -- Counts distinct movies
151 from rental
152 Join inventory ON rental.inventory_id = inventory.inventory_id -- Joins inventory and rental tables through inventory id
153 Join film ON inventory.film_id = film.film_id -- Joins film and inventory tables through film id
154 Join staff ON rental.staff_id = staff.staff_id -- Joins staff and rental tables through staff id
155 Where staff.first_name = 'Mike' -- Selects stores where the staff's name is Mike
156 And datediff(rental.rental_date, rental.return_date) = 10; -- Selects the movies from his store which were rented for 10 days
157
158 -- Query 6
159
160 • Select actor.actor_id, actor.last_name, actor.first_name -- Selects actors names and their id numbers
161 from actor
162 Join film_actor ON actor.actor_id = film_actor.actor_id -- Joins film-actor and actor tables by actor id
163 Where film_actor.film_id = ( -- Orders movies by the number of actors featured in them
164 Select film_id
165 from film_actor
166 Group by film_id
167 Order by count(actor_id) desc
168 limit 1
169 )
170 Order by actor.first_name, actor.last_name; -- Orders the output alphabetically by actor first name

```

The Result Grid shows the following data:

count(distinct film.film_id)
0

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

MySQL Workbench interface showing Query 6. The query is as follows:

```

150 • Select count(distinct film.film_id) -- Counts distinct movies
151 from rental
152 Join inventory ON rental.inventory_id = inventory.inventory_id -- Joins inventory and rental tables through inventory id
153 Join film ON inventory.film_id = film.film_id -- Joins film and inventory tables through film id
154 Join staff ON rental.staff_id = staff.staff_id -- Joins staff and rental tables through staff id
155 Where staff.first_name = 'Mike' -- Selects stores where the staff's name is Mike
156 And datediff(rental.rental_date, rental.return_date) = 10; -- Selects the movies from his store which were rented for 10 days
157
158 -- Query 6
159
160 • Select actor.actor_id, actor.last_name, actor.first_name -- Selects actors names and their id numbers
161 from actor
162 Join film_actor ON actor.actor_id = film_actor.actor_id -- Joins film-actor and actor tables by actor id
163 Where film_actor.film_id = ( -- Orders movies by the number of actors featured in them
164 Select film_id
165 from film_actor
166 Group by film_id
167 Order by count(actor_id) desc
168 limit 1
169 )
170 Order by actor.first_name, actor.last_name; -- Orders the output alphabetically by actor first name

```

The Result Grid shows the following data:

actor_id	last_name	first_name
75	POSEY	BLURT
111	ZELLWEGER	CAMERON
61	NEESON	CHRISTIAN
147	WINSLET	FAY
150	NOLTE	JAYNE

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 customer 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 film-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
```