DBMS HW #6 Write-Up

SQL Printouts:

```
1
2   /*********************************************************************
3    * NAME: Colleen Lemak
4    * CLASS: DBMS
5    * DATE: 17 November 2022
6    * HOMEWORK: #6
7    * DESCRIPTION: This file implements HW6 by structuring queries for desired output.
8    *********************************************************************/
9
10
11  -- TODO: Implement the queries for HW6 below. Include comments as
12  --       appropriate.
13
14
15  -- Question 1:
16  SELECT COUNT(*), MIN(length), MAX(length), AVG(length), COUNT(DISTINCT special_features)
17  FROM film;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)	MIN(length)	MAX(length)	AVG(length)	COUNT(DISTINCT special_features)
1000	46	185	115.2720	15

```
8   *********************************************************************/
9
10
11  -- TODO: Implement the queries for HW6 below. Include comments as
12  --       appropriate.
13
14
15  -- Question 1:
16  SELECT COUNT(*), MIN(length), MAX(length), AVG(length), COUNT(DISTINCT special_features)
17  FROM film;
18
19  -- Question 2:
20  SELECT rating, COUNT(*), ROUND(AVG(length), 2)
21  FROM film
22  GROUP BY rating
23  ORDER BY ROUND(AVG(length), 2) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

rating	COUNT(*)	ROUND(AVG(length), 2)
PG-13	223	120.44
R	195	118.66
NC-17	210	113.23
PG	194	112.01
G	178	111.05

```sql
-- Question 2:
SELECT rating, COUNT(*), ROUND(AVG(length), 2)
FROM film
GROUP BY rating
ORDER BY ROUND(AVG(length), 2) DESC;

-- Question 3:
SELECT rating, COUNT(*)
FROM actor  JOIN film_actor USING (actor_id) JOIN film USING (film_id)
GROUP BY rating
ORDER BY COUNT(*) DESC;


-- Question 4:
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

rating	COUNT(*)
PG-13	1184
PG	1143
NC-17	1128
R	1031
G	976

```sql
-- Question 4:
SELECT c.name, COUNT(*), MIN(rental_rate), MAX(rental_rate), ROUND(AVG(rental_rate), 2), MIN(replacement_cost), MAX(replacement_cost), ROUND
FROM category c JOIN film_category fc USING (category_id) JOIN film f USING (film_id)
GROUP BY c.name ASC;

-- Question 5:
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	COUNT(*)	MIN(rental_rate)	MAX(rental_rate)	ROUND(AVG(rental_rate), 2)	MIN(replacement_cost)	MAX(replacement_cost)	ROUND(AVG(re 2)
Action	64	0.99	4.99	2.65	9.99	29.99	20.91
Animation	66	0.99	4.99	2.81	9.99	29.99	20.13
Children	60	0.99	4.99	2.89	9.99	29.99	20.06
Classics	57	0.99	4.99	2.74	10.99	29.99	21.01
Comedy	58	0.99	4.99	3.16	9.99	28.99	19.02
Documentary	68	0.99	4.99	2.67	9.99	29.99	19.62
Drama	62	0.99	4.99	3.02	9.99	29.99	21.09
Family	69	0.99	4.99	2.76	9.99	29.99	19.73
Foreign	73	0.99	4.99	3.10	9.99	29.99	18.65
Games	61	0.99	4.99	3.25	9.99	29.99	20.29
Horror	56	0.99	4.99	3.03	10.99	29.99	19.87
Music	51	0.99	4.99	2.95	10.99	29.99	19.44
New	63	0.99	4.99	3.12	9.99	29.99	19.42
Sci-Fi	61	0.99	4.99	3.22	9.99	29.99	21.15
Sports	74	0.99	4.99	3.13	9.99	29.99	20.40
Travel	57	0.99	4.99	3.24	9.99	29.99	19.03

```sql
-- Question 5:
SELECT f.rating, COUNT(*)
FROM inventory i JOIN rental r USING (inventory_id) JOIN film f USING (film_id) JOIN film_category fc USING (film_id)
WHERE fc.category_id = 11 AND i.store_id = 1
GROUP BY f.rating
ORDER BY COUNT(*) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

rating	COUNT(*)
R	119
PG-13	102
PG	71
G	64
NC-17	30

```sql
-- Question 6:
SELECT f.title, f.rating, COUNT(*)
FROM rental r JOIN inventory i USING (inventory_id) JOIN film f USING (film_id) JOIN film_category fc USING (film_id)
WHERE fc.category_id = 11
GROUP BY f.title
ORDER BY COUNT(*) DESC
LIMIT 15;

-- Question 7:
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

title	rating	COUNT(*)
PULP BEVERLY	G	30
FAMILY SWEET	R	29
SWARM GOLD	PG-13	27
STREETCAR INTENTIONS	R	25
ARACHNOPHOBIA ROLLERCOASTER	PG-13	24
LOLA AGENT	PG	24
SLEEPING SUSPECTS	PG-13	24
YENTL IDAHO	R	23
PATTON INTERVIEW	PG	23
CARRIE BUNCH	PG	23
KARATE MOON	PG-13	23
AFFAIR PREJUDICE	G	23
FREDDY STORM	NC-17	23
HIGH ENCINO	R	22
SINNERS ATLANTIS	PG-13	22

```sql
-- Question 7:
SELECT f.title, COUNT(*)
FROM film f JOIN film_category fc USING (film_id) JOIN inventory i USING (film_id) JOIN rental r USING (inventory_id)
WHERE f.rating = "G" AND fc.category_id = 1 -- 1 is action film
GROUP BY f.title
HAVING COUNT(*) >= 15
ORDER BY COUNT(*) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	COUNT(*)
PRIMARY GLASS	27
HILLS NEIGHBORS	25
TRUMAN CRAZY	23
EASY GLADIATOR	23
EXCITEMENT EVE	21
BRIDE INTRIGUE	19
BAREFOOT MANCHURIAN	18
DARN FORRESTER	18
WATERFRONT DELIVERANCE	17
MIDNIGHT WESTWARD	17
LUST LOCK	15
RINGS HEARTBREAKERS	15

```sql
-- Question 8:
SELECT first_name, last_name, COUNT(*)
FROM actor a JOIN film_actor fa USING (actor_id) JOIN film_category fc USING (film_id)
WHERE fc.category_id = 11 -- 11 is horror film
GROUP BY a.actor_id
HAVING COUNT(*) >= 4
ORDER BY COUNT(*) DESC, a.last_name ASC, a.first_name ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	COUNT(*)
JULIA	MCQUEEN	7
TOM	MIRANDA	6
VIVIEN	BERGEN	5
HENRY	BERRY	5
KARL	BERRY	4
JOHNNY	CAGE	4
RIP	CRAWFORD	4
PENELOPE	CRONYN	4
JUDE	CRUISE	4
ED	GUINESS	4
WHOOPI	HURT	4
GRETA	MALDEN	4
PENELOPE	PINKETT	4
BURT	POSEY	4
KENNETH	TORN	4
WALTER	TORN	4

```sql
-- Question 9:
SELECT c.first_name, c.last_name, COUNT(*)
FROM customer c JOIN rental r USING (customer_id) JOIN inventory i USING (inventory_id) JOIN film f USING (film_id)
WHERE f.rating = "PG"
GROUP BY c.customer_id
HAVING COUNT(*) > 10
ORDER BY COUNT(*) DESC, c.last_name ASC, c.first_name ASC;

-- Question 10:

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	COUNT(*)
DUSTIN	GILLETTE	13
TAMMY	SANDERS	13
JUNE	CARROLL	12
ELEANOR	HUNT	12
BILLY	POULIN	12
TERRANCE	ROUSH	12
CLARA	SHAW	12
JULIA	FLORES	11
ELSIE	KELLEY	11
BECKY	MILES	11
SUE	PETERS	11
SALLY	PIERCE	11
LESLIE	SEWARD	11

Result 29 | Read Only

Limit to 1000 rows

```sql
-- Question 10:
/*
Exploratory Analysis Query:
I want to find the top 10 returning and loyal customers who have paid the most amount in a single transaction.
This means the top 10 paying customers will only be selected if they have made more than 20 payments.
The returned data will be sorted from highest to lowest amounts paid, the corresponding customer first name then last name, alphabetically,
followed by date.
The objective is to find the most loyal customer with high payments.
*/
SELECT c.first_name, c.last_name, p.amount, AVG(p.amount), p.payment_date, COUNT(*)
FROM customer c JOIN payment p USING (customer_id)
GROUP BY c.customer_id
HAVING COUNT(*) > 20
ORDER BY p.amount DESC, c.last_name ASC, c.first_name ASC, p.payment_date DESC
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

first_name	last_name	amount	AVG(p.amount)	payment_date	COUNT(*)
ALMA	AUSTIN	11.99	4.332857	2005-05-25 18:18:19	35
RITA	GRAHAM	10.99	3.865000	2005-05-26 02:26:23	24
DAVID	ROYAL	10.99	4.451538	2005-05-25 21:58:58	26
TONYA	CHAPMAN	9.99	5.052500	2005-05-27 03:55:25	32
JOHNNIE	CHISHOLM	9.99	5.073333	2005-05-26 10:54:28	24
JEANNE	LAWSON	9.99	5.064074	2005-05-26 16:20:56	27
EVELYN	MORGAN	9.99	4.097143	2005-05-26 00:41:10	28
PEGGY	MYERS	9.99	4.031667	2005-05-27 21:13:10	24
ANNE	POWELL	9.99	3.816087	2005-05-29 00:54:53	23
KELLY	TORRES	9.99	4.535455	2005-05-27 02:22:26	22

Question 10 Explanation:

I find this query interesting because we are able to group on customer id, meaning we can count up all of that customer's payments, and investigate who is most important to the company financially, for example. Reporting the top 10 most engaged customers (via payments) is important because you want to understand who is paying the most, especially if their payments are frequent. Perhaps we want to reward the top 10 customers with bonuses to encourage their return. Averaging their payment amount, especially if applied to a very large database can provide insight into which customers are most active.