

Assignment 1

1. Display the first and last name of each actor in a single column in upper case letters in alphabetic order. Name the column Actor Name.

```
select upper(concat(first_name,' ',last_name)) as 'Actor_name' from actor order by first_name;
```

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL statement:

```
1 • select upper(concat(first_name,' ',last_name)) as 'Actor_name' from actor order by first_name;
2
```

The result grid displays the output of the query, showing a single column named 'Actor_name' with 20 rows of actor names in uppercase, sorted alphabetically by first name.

Actor_name
ADAM GRANT
ADAM HOPPER
AL GARLAND
ALAN DREYFUSS
ALBERT NOLTE
ALBERT JOHANSSON
ALEC WAYNE
ANGELA HUDSON
ANGELA WITHERSPOON
ANGELINA ASTAIRE
ANNE CRONYN
AUDREY BAILEY
AUDREY OLIVIER
BELA WALKEN
BEN HARRIS
BEN WILLIS
BETTE NICHOLSON
BOB FAWCETT
BURT DUKAKIS
BURT POSEY
BURT TEMPLE
CAMERON WADY

2. Find all actors whose last name contain the letters GEN:

```
select concat(first_name,' ',last_name) as 'Actor_name'
from actor
where last_name like '%Gen%';
```

employees_audit employees_audit employees_audit employees_audit - Table SQL File 18 task task_rentdvd SQL

Limit to 10000 rows

```

1 • select upper(concat(first_name, ' ', last_name)) as 'Actor_name' from actor order by first_name;
2 • select concat(first_name, ' ', last_name) as 'Actor_name'
3   from actor
4  where last_name like '%Gen%';
5

```

Result Grid Filter Rows: Export: Wrap Cell Content:

Actor_name
VIVIEN BERGEN
JODIE DEGENERES
GINA DEGENERES
NICK DEGENERES

3. Using IN, display the country_id and country columns of the following countries:
Afghanistan, Bangladesh, and China:

`select country_id, country from country where country in ('Afghanistan', 'Bangladesh', 'China');`

employees_audit employees_audit employees_audit employees_audit - Table SQL File 18 task task_rentdvd

Limit to 10000 rows

```

1 • select upper(concat(first_name, ' ', last_name)) as 'Actor_name' from actor order by first_name;
2 • select concat(first_name, ' ', last_name) as 'Actor_name'
3   from actor
4  where last_name like '%Gen%';
5 • select country_id, country
6   from country
7  where country in ('Afghanistan', 'Bangladesh', 'China');

```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:

country_id	country
1	Afghanistan
12	Bangladesh
23	China
NULL	NULL

4. List the last names of actors, as well as how many actors have that last name.

`select last_name, count(last_name) from actor group by last_name;`

employees_audit employees_audit employees_audit employees_audit - Table SQL File 18 task

Limit to 10000 rows

```

9 • select last_name, count(last_name)
10   from actor group by last_name;
11
12
13
14
15
16

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	last_name	count(last_name)
▶	AKROYD	3
	ALLEN	3
	ASTAIRE	1
	BACALL	1
	BAILEY	2
	BALE	1
	BALL	1
	BARRYMORE	1
	BASINGER	1
	BENING	2
	BERGEN	1
	BERGMAN	1
	BERRY	3
	BIRCH	1
	BLOOM	1
	BOLGER	2

5. List last names of actors and the number of actors who have that last name, but only for names that are shared by at least two actors

`select last_name, count(last_name) as 'CNT' from actor group by last_name having CNT > 2;`

employees_audit employees_audit employees_audit employees_audit - Table SQL File 18

Limit to 10000 rows

```

11
12 • select last_name, count(last_name) as 'CNT'
13   from actor group by last_name having CNT > 2;
14
15
16
17
18

```

Result Grid Filter Rows: Export: Wrap Cell Content:

	last_name	CNT
▶	AKROYD	3
	ALLEN	3
	BERRY	3
	DAVIS	3
	DEGENERES	3
	GARLAND	3
	GUINNESS	3
	HARRIS	3
	HOFFMAN	3
	HOPKINS	3
	JOHANSSON	3
	KEITEL	3
	KILMER	5
	NOLTE	4
	PECK	3
	TEMPLE	4
	TORN	3
	WILLIAMS	3
	WILLIS	3
	ZELLWEGER	3

6. The actor HARPO WILLIAMS was accidentally entered in the actor table as GROUCHO WILLIAMS. Write a query to fix the record.

```
update actor set first_name = 'HARPO' where first_name = 'GROUCHO' and last_name = 'WILLIAMS';
```

The screenshot shows a SQL IDE window with the following tabs: employees_audit, employees_audit, employees_audit, employees_audit - Table, SQL File 18, task, and ta. The SQL editor contains the following code:

```
15
16 • update actor set first_name = 'HARPO'
17   where first_name = 'GROUCHO' and last_name = 'WILLIAMS';
18
19 • select first_name, last_name
20   from actor
21   where first_name = 'HARPO' and last_name = 'WILLIAMS';
22
```

Below the editor, the 'Result Grid' tab is active, showing the results of the SELECT query:

first_name	last_name
HARPO	WILLIAMS

7. Use JOIN to display the first and last names, as well as the address, of each staff member. Use the tables staff and address:

```
select s.first_name, s.last_name ,a.address as 'Staff details' from staff as s join address a on s.address_id = a.address_id;
```

The screenshot shows a SQL IDE window with the following tabs: employees_audit, employees_audit, employees_audit, employees_audit - Table, SQL File 18, and 1. The SQL editor contains the following code:

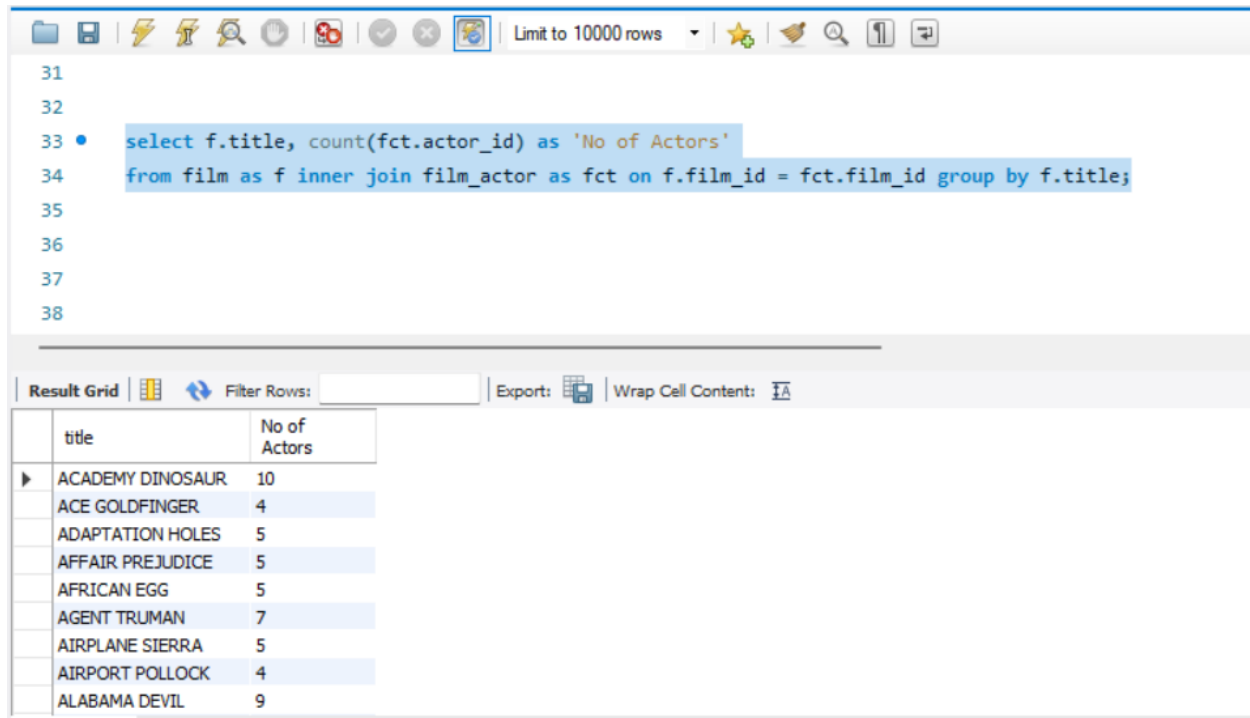
```
23
24
25 • select s.first_name, s.last_name ,a.address as 'Staff details'
26   from
27   staff as s
28   join
29   address a
30   on s.address_id = a.address_id;
```

Below the editor, the 'Result Grid' tab is active, showing the results of the JOIN query:

first_name	last_name	Staff details
Mike	Hillyer	23 Workhaven Lane
Jon	Stephens	1411 Lillydale Drive

8. List each film and the number of actors who are listed for that film. Use tables film_actor and film. Use inner join.

```
select f.title, count(fct.actor_id) as 'No of Actors' from film as f inner join film_actor as fct on f.film_id = fct.film_id group by f.title;
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, a 'Limit to 10000 rows' dropdown, and a star icon. The SQL editor contains the following query:

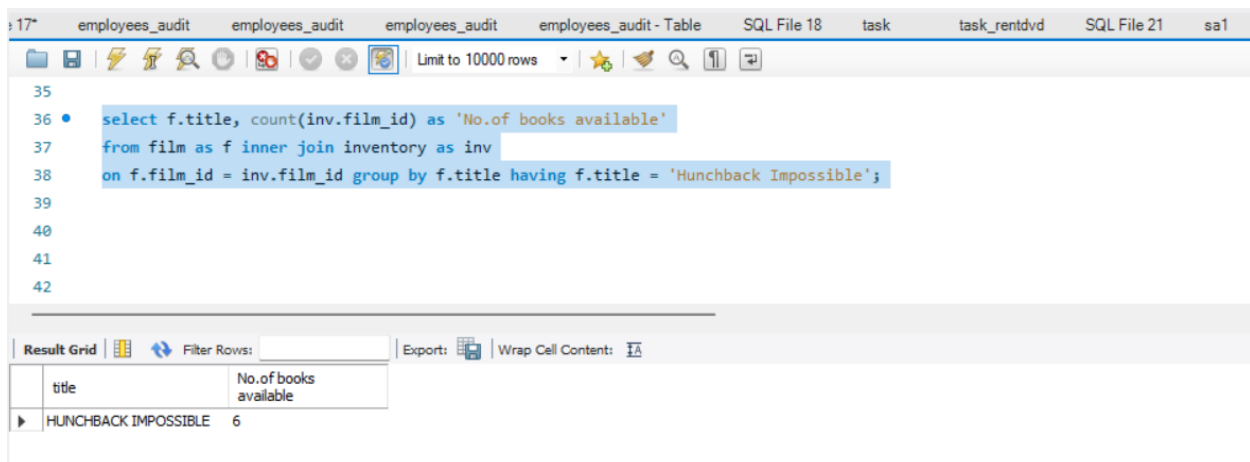
```
31
32
33 • select f.title, count(fct.actor_id) as 'No of Actors'
34   from film as f inner join film_actor as fct on f.film_id = fct.film_id group by f.title;
35
36
37
38
```

Below the editor is the 'Result Grid' section. It has a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with two columns: 'title' and 'No of Actors'.

title	No of Actors
ACADEMY DINOSAUR	10
ACE GOLDFINGER	4
ADAPTATION HOLES	5
AFFAIR PREJUDICE	5
AFRICAN EGG	5
AGENT TRUMAN	7
AIRPLANE SIERRA	5
AIRPORT POLLOCK	4
ALABAMA DEVIL	9

9. How many copies of the film Hunchback Impossible exist in the inventory system?

```
select f.title, count(inv.film_id) as 'No.of books available' from film as f inner join inventory as inv on f.film_id = inv.film_id group by f.title having f.title = 'Hunchback Impossible';
```



The screenshot shows a SQL IDE interface with multiple tabs open: 'employees_audit', 'employees_audit - Table', 'SQL File 18', 'task', 'task_rentdvd', 'SQL File 21', and 'sa1'. The SQL editor contains the following query:

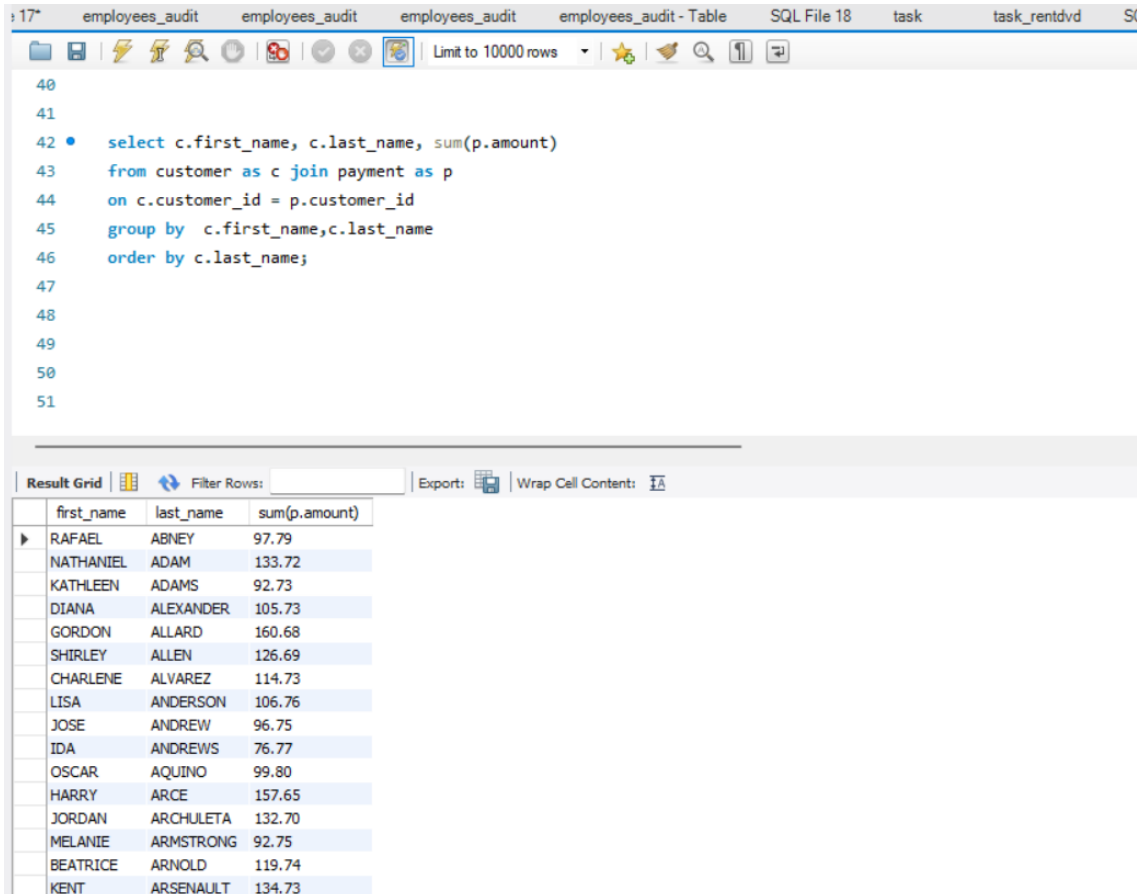
```
35
36 • select f.title, count(inv.film_id) as 'No.of books available'
37   from film as f inner join inventory as inv
38   on f.film_id = inv.film_id group by f.title having f.title = 'Hunchback Impossible';
39
40
41
42
```

Below the editor is the 'Result Grid' section. It has a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with two columns: 'title' and 'No.of books available'.

title	No.of books available
HUNCHBACK IMPOSSIBLE	6

10. Using the tables payment and customer and the JOIN command, list the total paid by each customer. List the customers alphabetically by last name

`select c.first_name, c.last_name, sum(p.amount) from customer as c join payment as p on c.customer_id = p.customer_id group by c.first_name, c.last_name order by c.last_name;`



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```

40
41
42 • select c.first_name, c.last_name, sum(p.amount)
43 from customer as c join payment as p
44 on c.customer_id = p.customer_id
45 group by c.first_name, c.last_name
46 order by c.last_name;
47
48
49
50
51

```

The result grid displays the following data:

first_name	last_name	sum(p.amount)
RAFAEL	ABNEY	97.79
NATHANIEL	ADAM	133.72
KATHLEEN	ADAMS	92.73
DIANA	ALEXANDER	105.73
GORDON	ALLARD	160.68
SHIRLEY	ALLEN	126.69
CHARLENE	ALVAREZ	114.73
LISA	ANDERSON	106.76
JOSE	ANDREW	96.75
IDA	ANDREWS	76.77
OSCAR	AQUINO	99.80
HARRY	ARCE	157.65
JORDAN	ARCHULETA	132.70
MELANIE	ARMSTRONG	92.75
BEATRICE	ARNOLD	119.74
KENT	ARSENAULT	134.73

11. The music of Queen and Kris Kristofferson have seen an unlikely resurgence. As an unintended consequence, films starting with the letters K and Q have also soared in popularity. Use subqueries to display the titles of movies starting with the letters K and Q whose language is English.

`select f.title from film as f where f.language_id in (select lng.language_id from language lng where name = 'English') and f.title rlike '^[K,Q]';`

employees_audit employees_audit employees_audit employees_audit - Table SQL File 18

Limit to 10000 rows

```

47
48
49 • select f.title from film as f
50 where f.language_id in
51 (select lng.language_id from language lng where name = 'English')
52 and f.title rlike '^[K,Q]';
53

```

Result Grid Filter Rows: Export: Wrap Cell Content:

title
KANE EXORCIST
KARATE MOON
KENTUCKIAN GIANT
KICK SAVANNAH
KILL BROTHERHOOD
KILLER INNOCENT
KING EVOLUTION
KISS GLORY
KISSING DOLLS
KNOCK WARLOCK
KRAMER CHOCOLATE
KWAI HOMEWARD
QUEEN LUKE
QUEST MUSSOLINI
QUILLS BULL

12. Use subqueries to display all actors who appear in the film *Alone Trip*.

```

select a.first_name,a.last_name from actor as a where a.actor_id in (select fm.actor_id
from film_actor as fm where fm.film_id in (select f.film_id from film as f where
f.title='Alone Trip'));

```

employees_audit employees_audit employees_audit employees_audit - Table SQL File 18 task task_rentdvd SC

Limit to 10000 rows

```

53
54
55
56
57 • select a.first_name,a.last_name from actor as a where a.actor_id in
58 (select fm.actor_id from film_actor as fm where fm.film_id in
59 (select f.film_id from film as f where f.title='Alone Trip'));
60

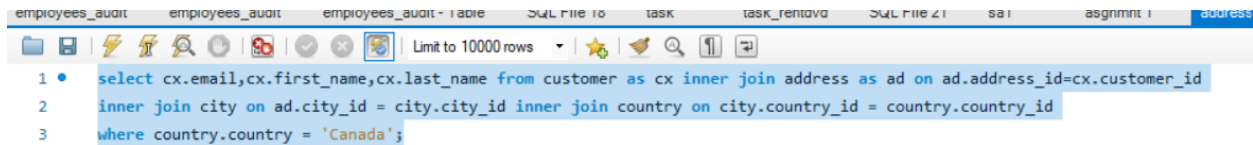
```

Result Grid Filter Rows: Export: Wrap Cell Content:

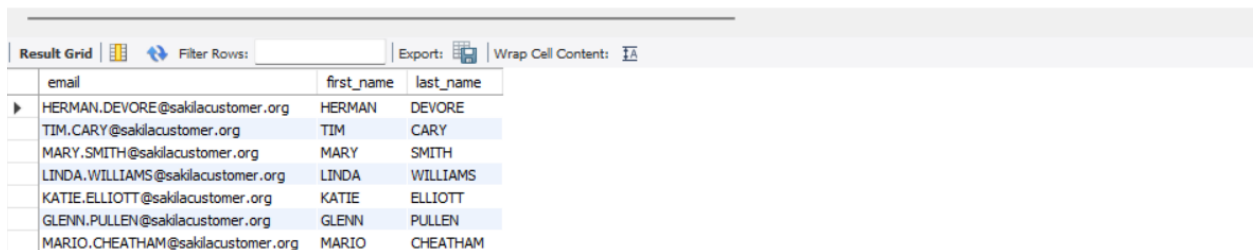
first_name	last_name
ED	CHASE
KARL	BERRY
UMA	WOOD
WOODY	JOLIE
SPENCER	DEPP
CHRIS	DEPP
LAURENCE	BULLOCK
RENEE	BALL

13. You want to run an email marketing campaign in Canada, for which you will need the names and email addresses of all Canadian customers. Use joins to retrieve this information.

```
select cx.email,cx.first_name,cx.last_name from customer as cx inner join address as ad on ad.address_id=cx.customer_id inner join city on ad.city_id = city.city_id inner join country on city.country_id = country.country_id where country.country = 'Canada';
```



The screenshot shows a SQL IDE window with a query editor. The query is: `select cx.email,cx.first_name,cx.last_name from customer as cx inner join address as ad on ad.address_id=cx.customer_id inner join city on ad.city_id = city.city_id inner join country on city.country_id = country.country_id where country.country = 'Canada';`. The query is highlighted in blue. The IDE has a toolbar with various icons and a status bar at the bottom.



The screenshot shows the result grid of the SQL IDE. The result grid has 7 rows and 3 columns: email, first_name, and last_name. The data is as follows:

email	first_name	last_name
HERMAN.DEVORE@sakilacustomer.org	HERMAN	DEVORE
TIM.CARY@sakilacustomer.org	TIM	CARY
MARY.SMITH@sakilacustomer.org	MARY	SMITH
LINDA.WILLIAMS@sakilacustomer.org	LINDA	WILLIAMS
KATIE.ELLIOTT@sakilacustomer.org	KATIE	ELLIOTT
GLENN.PULLEN@sakilacustomer.org	GLENN	PULLEN
MARIO.CHEATHAM@sakilacustomer.org	MARIO	CHEATHAM

14. Sales have been lagging among young families, and you wish to target all family movies for a promotion. Identify all movies categorized as family films.

```
select f.title from film as f join film_category as fcat on f.film_id=fcat.film_id inner join category on category.category_id=fcat.category_id where name='Family';
```


uditemployees_auditemployees_audit - TableSQL File 18tasktask_rentdvdSQL File 21sa1asgnmnt 1*addresscountryc

Limit to 10000 rows

</

15. Create a Stored procedure to get the count of films in the input category (IN category_name, OUT count)

delimiter \$\$

```
CREATE PROCEDURE pr_filmcnt(IN cat_name VARCHAR(255), OUT cnt INT)
BEGIN
```

```
    SELECT COUNT(*) INTO cnt FROM film_category as fc
    JOIN category c ON fc.category_id = c.category_id
    WHERE c.name = cat_name;
```

```
END $$
```

delimiter ;

```
call pr_filmcnt('Family',@cnt);
```

```
select @cnt;
```

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to "Limit to 10000 rows". The SQL editor contains the following code:

```
1 delimiter $$
2
3 CREATE PROCEDURE pr_filmcnt(IN cat_name VARCHAR(255), OUT cnt INT)
4 BEGIN
5     SELECT COUNT(*) INTO cnt FROM film_category as fc
6     JOIN category c ON fc.category_id = c.category_id
7     WHERE c.name = cat_name;
8 END $$
9 delimiter ;
10 call pr_filmcnt('Family',@cnt);
11 select @cnt;
```

Below the editor is the "Result Grid" section. It has a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The result grid displays a single row with the value 69 for the variable @cnt.

@cnt
69

16. Display the most frequently rented movies in descending order.

```
select title, count(rental_id) as 'mostrentedcount'
from rental
join inventory
on (rental.inventory_id = inventory.inventory_id)
join film
on (inventory.film_id = film.film_id)
group by film.title
order by count(rental_id) desc;
```

Limit to 10000 rows

```

85 • SELECT title, COUNT(rental_id) as 'mostrentedcount'
86 FROM rental
87 JOIN inventory
88 ON (rental.inventory_id = inventory.inventory_id)
89 JOIN film
90 ON (inventory.film_id = film.film_id)
91 GROUP BY film.title
92 ORDER BY COUNT(rental_id) DESC;
93

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [fA](#)

	title	mostrentedcount
▶	BUCKET BROTHERHOOD	34
	ROCKETEER MOTHER	33
	FORWARD TEMPLE	32
	GRIT CLOCKWORK	32
	JUGGLER HARDLY	32
	RIDGEMONT SUBMARINE	32
	SCALAWAG DUCK	32
	APACHE DIVINE	31
	GOODFELLAS SALUTE	31
	HOBBIT ALIEN	31
	NETWORK PEAK	31
	DORRERS TOWN	31

17. Write a query to display for each store its store ID, city, and country.

```

SELECT
    s.store_id, c.city, cn.country
FROM
    store AS s
    INNER JOIN
    address AS ad ON s.address_id = ad.address_id
    JOIN
    city AS c
on
    c.city_id=ad.city_id
    inner join country cn on cn.country_id=c.country_id;

```

outine SQL File 17* SQL File 18 task task_rentdvd SQL File 21 sa1 asgnmnt 1* x ad

Limit to 10000 rows

```

94 • SELECT
95     s.store_id, c.city, cn.country
96 FROM
97     store AS s
98     INNER JOIN
99     address AS ad ON s.address_id = ad.address_id
100    JOIN
101    city AS c
102    on
103    c.city_id=ad.city_id
104    inner join country cn on cn.country_id=c.country_id;

```

Result Grid

	store_id	city	country
	1	Lethbridge	Canada
▶	2	Woodridge	Australia

18. List the genres and its gross revenue.

```

select ctg.name,sum(p.amount) as 'Gross Revenue' from category ctg inner join
film_category fc on ctg.category_id = fc.category_id
inner join inventory i on fc.film_id = i.film_id inner join rental r on i.inventory_id =
r.inventory_id inner join payment p on r.rental_id = p.rental_id group by ctg.name order
by sum(p.amount);

```

The screenshot shows a SQL IDE with a query editor and a results grid. The query editor contains the following SQL code:

```
111 • select ctg.name,sum(p.amount) as 'Gross Revenue' from category ctg inner join film_category fc on ctg.category_id = fc.category_id
112 inner join inventory i on fc.film_id = i.film_id inner join rental r on i.inventory_id = r.inventory_id
113 inner join payment p on r.rental_id = p.rental_id group by ctg.name order by sum(p.amount) desc limit 5;
```

The results grid displays the following data:

name	Gross Revenue
Sports	5314.21
Sci-Fi	4756.98
Animation	4656.30
Drama	4587.39
Comedy	4383.58

19. Create a View for the above query(18)

create view top5 as

```
select ctg.name,sum(p.amount) as 'Gross Revenue' from category ctg inner join
film_category fc on ctg.category_id = fc.category_id
inner join inventory i on fc.film_id = i.film_id inner join rental r on i.inventory_id =
r.inventory_id inner join payment p on r.rental_id = p.rental_id group by ctg.name order
by sum(p.amount) desc limit 5;
```

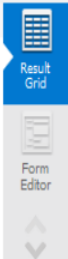
```

16
17 • create view top5 as
18   select ctg.name, sum(p.amount) as 'Gross Revenue' from category ctg inner join film_category fc on ctg.category_id = fc.category_id
19   inner join inventory i on fc.film_id = i.film_id inner join rental r on i.inventory_id = r.inventory_id
20   inner join payment p on r.rental_id = p.rental_id group by ctg.name order by sum(p.amount) desc limit 5;
21
22 • select * from top5
23
24

```

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

name	Gross Revenue
Sports	5314.21
Sci-Fi	4756.98
Animation	4656.30
Drama	4587.39
Comedy	4383.58



5 25 x Read Only Con

20. Select top 5 genres in gross revenue view.

select * from top5

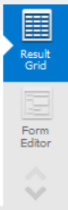
```

16
17 • create view top5 as
18   select ctg.name, sum(p.amount) as 'Gross Revenue' from category ctg inner join film_category fc on ctg.category_id = fc.category_id
19   inner join inventory i on fc.film_id = i.film_id inner join rental r on i.inventory_id = r.inventory_id
20   inner join payment p on r.rental_id = p.rental_id group by ctg.name order by sum(p.amount) desc limit 5;
21
22 • select * from top5
23
24

```

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

name	Gross Revenue
Sports	5314.21
Sci-Fi	4756.98
Animation	4656.30
Drama	4587.39
Comedy	4383.58



5 25 x Read Only Con