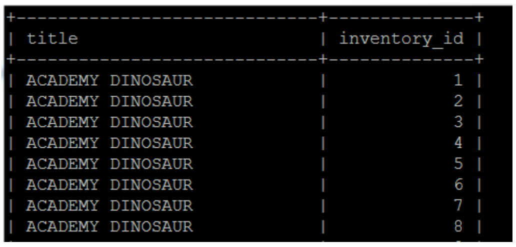
Joins Exercise 6-1

Using sakila database - Checkpoint Lab

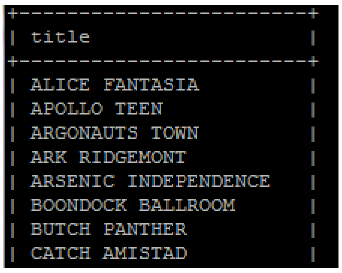
1. Get a list of all film titles and their inventory number – even those we don't have.
   - ➢ You might like to limit the output to 200 to check you have everything looking like this



```
+----------------------------+-------------+
| title                      | inventory_id |
+----------------------------+-------------+
| ACADEMY DINOSAUR           |           1 |
| ACADEMY DINOSAUR           |           2 |
| ACADEMY DINOSAUR           |           3 |
| ACADEMY DINOSAUR           |           4 |
| ACADEMY DINOSAUR           |           5 |
| ACADEMY DINOSAUR           |           6 |
| ACADEMY DINOSAUR           |           7 |
| ACADEMY DINOSAUR           |           8 |
```

select film.title, inventory.inventory_id from film join inventory on film.film_id=inventory.film_id LIMIT 200;

2. Which films do we ***not*** have in stock?
   - ➢ There are 42 records beginning with:

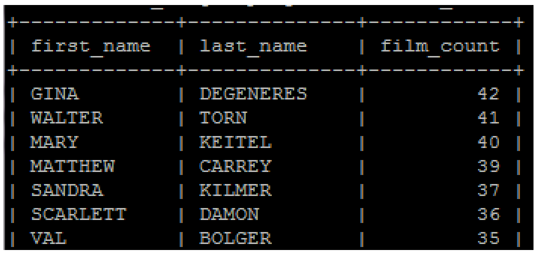


```
+-------------------------+
| title                   |
+-------------------------+
| ALICE FANTASIA          |
| APOLLO TEEN             |
| ARGONAUTS TOWN          |
| ARK RIDGEMONT           |
| ARSENIC INDEPENDENCE    |
| BOONDOCK BALLROOM       |
| BUTCH PANTHER           |
| CATCH AMISTAD           |
```

MariaDB [sakila]> select film.title from film left join inventory on film.film_id=inventory.film_id where inventory.inventory_id IS NULL;

3. List the number of films in which each actor has featured (sort the output in descending order of the number of films)

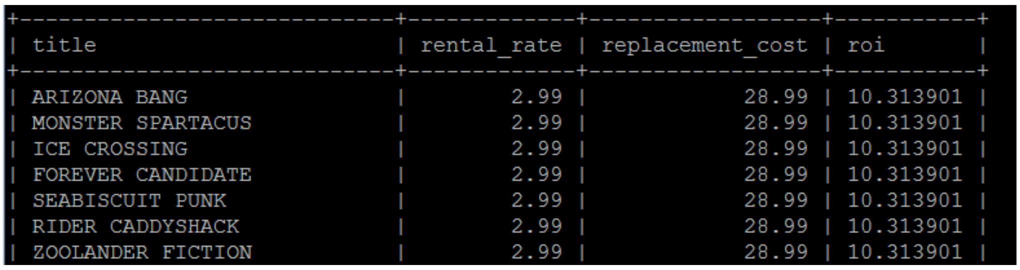
select actor.first_name, actor.last_name, count(film_actor.film_id) AS film_count from actor join film_actor on actor.actor_id=film_actor.actor_id group by actor.actor_id ORDER BY film_count DESC;



```
+-------------+--------------+------------+
| first_name  | last_name    | film_count |
+-------------+--------------+------------+
| GINA        | DEGENERES    |         42 |
| WALTER      | TORN         |         41 |
| MARY        | KEITEL       |         40 |
| MATTHEW     | CARREY       |         39 |
| SANDRA      | KILMER       |         37 |
| SCARLETT    | DAMON        |         36 |
| VAL         | BOLGER       |         35 |
```

Techniques from this week

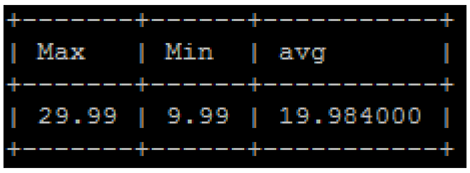
4. The store uses a formula to calculate the return-on-investment (or ROI) which is (rental_rate / replacement_cost * 100). List the films, rental replacement cost and ROI which have an ROI more than 10. Order by ROI. ***Only have the formula once in the query***



```
+-----------------------------+-------------+------------------+-----------+
| title                       | rental_rate | replacement_cost | roi       |
+-----------------------------+-------------+------------------+-----------+
| ARIZONA BANG                |        2.99 |            28.99 | 10.313901 |
| MONSTER SPARTACUS           |        2.99 |            28.99 | 10.313901 |
| ICE CROSSING                |        2.99 |            28.99 | 10.313901 |
| FOREVER CANDIDATE           |        2.99 |            28.99 | 10.313901 |
| SEABISCUIT PUNK             |        2.99 |            28.99 | 10.313901 |
| RIDER CADDYSHACK            |        2.99 |            28.99 | 10.313901 |
| ZOOLANDER FICTION           |        2.99 |            28.99 | 10.313901 |
```

```
select title, rental_rate, replacement_cost, ROI
    -> FROM ( select title, rental_rate,replacement_cost, (rental_rate/replacement_cost) *100 AS ROI FROM film ) AS calculated_roi where ROI>10 ORDER BY ROI DESC;
```

5. List the maximum, minimum and average film replacement cost using subselects in the select clause only (do not use a FROM clause in the main query) – yes this is silly.



```
+-------+------+-----------+
| Max   | Min  | avg       |
+-------+------+-----------+
| 29.99 | 9.99 | 19.984000 |
+-------+------+-----------+
```

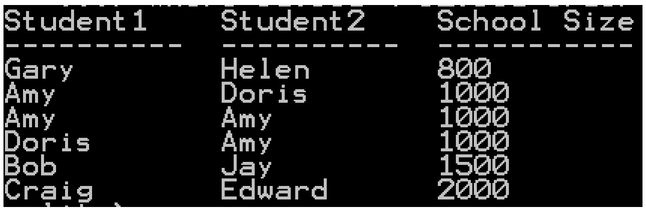
```
SELECT
    (SELECT MAX(replacement_cost) FROM film) AS max_replacement_cost,
    (SELECT MIN(replacement_cost) FROM film) AS min_replacement_cost,
    (SELECT AVG(replacement_cost) FROM film) AS avg_replacement_cost;
```

Student Database on SQLite

Techniques from earlier this week

Not trivial, you'll have to work on these.

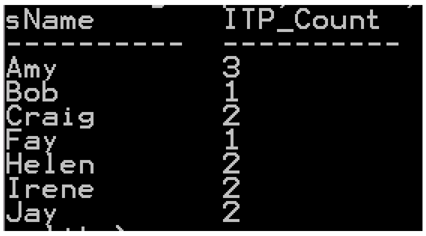
6. List the students as pairs who come from the same sized high school. Order by school size.
    - Only list one pair of each student e.g. if you have Alice and Bob in a record don't also list Bob and Alice (unless they are different students – we have two different AMY's).
    - Work through this in stages – removing redundant pairs is the last step. You might like to display more information while developing the query (e.g. sid)



```
Student1    Student2    School Size
----------  ----------  -----------
Gary        Helen       800
Amy         Doris       1000
Amy         Amy         1000
Doris       Amy         1000
Bob         Jay         1500
Craig       Edward      2000
```

```
sqlite> select  s1.sName, s2.sName, s1.sizeHS from student s1 join student s2
ON s1.sizeHS=s2.sizeHS AND s1.sID<s2.sID  ORDER BY s1.sizeHS;
```

7. List each student that has made an application and the number of ITP's they have applied to.
    - This is a simple inner join but I did use something introduced in passing today to get this.

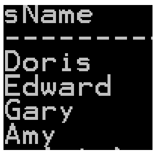


```
sName       ITP_Count
----------  ----------
Amy         3
Bob         1
Craig       2
Fay         1
Helen       2
Irene       2
Jay         2
```

```
sqlite> SELECT DISTINCT student.sName, (SELECT COUNT ( DISTINCT apply.itpName) FROM apply
where apply.sID=student.sID)AS ITP_count FROM student INNER JOIN apply on student.sID=apply.sID;
```

Subqueries and Outer Joins

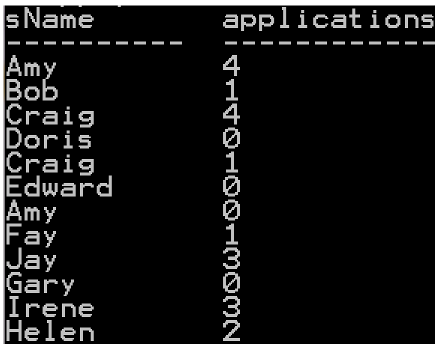
8. Which students have not applied anywhere?



```
sName
--------
Doris
Edward
Gary
Amy
```

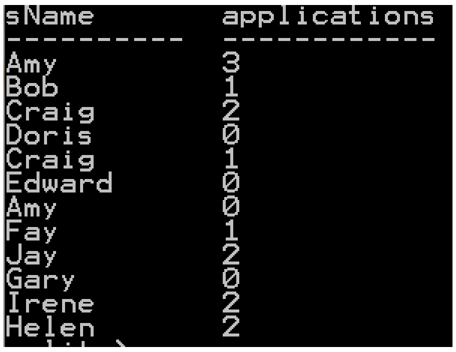
```
sqlite> select sName from student where sID NOT IN (select sID from apply);
sqlite> SELECT student.sName, (SELECT COUNT ( apply.itpName) FROM apply
where apply.sID=student.sID)AS ITP_count FROM student LEFT JOIN apply on student.sID=apply.sID group by student.sID;
```

9. List a count of the number of applications made by each student



```
sName       applications
----------  ------------
Amy         4
Bob         1
Craig       4
Doris       0
Craig       1
Edward      0
Amy         0
Fay         1
Jay         3
Gary        0
Irene       3
Helen       2
```

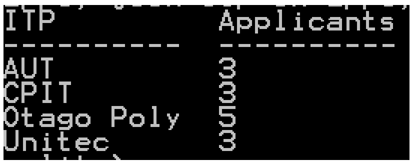
10. List the number of institutions that each student has applied to:



```
sName       applications
----------  ------------
Amy         3
Bob         1
Craig       2
Doris       0
Craig       1
Edward      0
Amy         0
Fay         1
Jay         2
Gary        0
Irene       2
Helen       2
```

```
SELECT student.sName, (SELECT COUNT (DISTINCT apply.itpName)
FROM apply where apply.sID=student.sID)AS ITP_count FROM student LEFT JOIN a
pply on student.sID=apply.sID group by student.sID;
```

11. How many students have applied to each institution?



```
ITP         Applicants
----------  ----------
AUT         3
CPIT        3
Otago Poly  5
Unitec      3
```

```
sqlite> select itpName, COUNT(DISTINCT sID) AS Applicants from apply GROUP BY itpName;
```