```
#### 4f. BETWEEN - SELECT rows with values in a range
    SELECT * FROM orders
    WHERE order_time BETWEEN '2017-01-01' AND '2017-01-07' ;

    IF dealing with int, no need of ' ' from range
```

```
204 •   SELECT * FROM orders
205       WHERE order_time BETWEEN '2017-01-01' AND '2017-01-07' ;
206
207 •     SELECT * FROM customers
208       WHERE last_name  BETWEEN 'A' AND 'L';
```

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

id	first_name	last_name	gender	phone_number
6	Katie	Armstrong	F	01145787353
7	Michael	Bluth	M	01980289282
9	Buster	Bluth	M	01173456782
11	Lindsay	Bluth	F	01176923804
12	Harry	Johnson	M	NULL
16	Gob	Bluth	M	01176985498
17	George	Bluth	M	01176984303
18	Lucille	Bluth	F	01198773214

customers 88

4e LIKE

```
SELECT * FROM cutomers WHERE firs_name LIKE '%da%';     - name contains 'da'
    'da%'        - name starts with 'da'
    '%da'        - name ends with 'da'
    '_o_'        - have just one ch on either side of o, EG- Gob
    SELECT title, stock_quantity FROM books WHERE stock_quantity LIKE '____';
    '%\%%';      - to detect '%'
    '%\_%';      = to detect '_'
```

```
SELECT * FROM products
WHERE price LIKE '3%';
```

Result Grid

id	name	price	coffee_origin
2	Macchiato	3.00	Colombia
3	Cappuccino	3.50	Costa Rica
4	Latte	3.50	Indonesia
5	Americano	3.25	Ethiopia
6	Flat White	3.50	Indonesia
7	Filter	3.00	Sri Lanka

```
-- last name to end with w
SELECT * FROM customers
WHERE last_name LIKE "%w";
```

Result Grid

id	first_name	last_name	gender	phone_number
2	Emma	Law	F	01123439899
NULL	NULL	NULL	NULL	NULL

```
#### 4g. ORDERBY ASC/DESC - by def it is ASC

    SELECT * FROM products
    ORDER BY price ASC;  - sim use DESC for descending
```

```
SELECT * FROM customers
ORDER BY last_name DESC;
```

Result Grid

id	first_name	last_name	gender	phone_number
4	Daniel	Williams	M	NULL
23	Toby	West	M	01176009822
3	Mark	Watkins	M	01174592013
5	Sarah	Taylor	F	01176348290
14	John	Taylor	M	NULL
13	John	Smith	M	01174987221
15	Emma	Smith	F	01176984116
21	John	Smith	M	01144473330

```
219 •        SELECT * FROM products
220        ORDER BY price ASC;
```

Result Grid | Filter Rows: | Edit:

id	name	price
3	BLack	1.50
4	latte	1.50
1	Espresso	3.50
2	Espresso	3.50
5	machi	3.50
6	crude	4.50
NULL	NULL	NULL

EX2.

Exercise 2

1. From the products table, select the name and price of all products with a coffee origin equal to Colombia or Indonesia. Ordered by name from A-Z.

2. From the orders table, select all the orders from February 2017 for customers with id's of 2, 4, 6 or 8.

3. From the customers table, select the first name and phone number of all customers who's last name contains the pattern 'ar'.

1. **SELECT name,price FROM products**
   WHERE coffe_origin IN (Colombia', 'Indonesia')
   ORDER BY name;

2. **SELECT * FROM orders**
   WHERE order_time BETWEEN '2017-02-01' AND '2017-02-28';
   AND customer_id IN(2,4,6,8);

```
SELECT * FROM orders
WHERE order_time BETWEEN '2017-02-01' AND '2017-02-28'
AND customer_id IN(2,4,6,8);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

id	product_id	customer_id	order_time
68	5	4	2017-02-10 11:05:34
69	1	2	2017-02-11 08:00:38
70	3	8	2017-02-12 08:08:08
73	5	2	2017-02-13 12:34:56
75	5	4	2017-02-14 09:12:56
78	1	8	2017-02-15 09:27:50
89	4	6	2017-02-20 10:43:39
92	3	2	2017-02-21 11:08:45

orders 98

3. **SELECT first_name, phone_number FROM customers**
   WHERE last_name LIKE '%ar%';

```
SELECT first_name, phone_number FROM customers
WHERE last_name LIKE '%ar%';
```

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

first_name	phone_number
Chris	01123147789
Katie	01145787353

```
#### 4h. DISTINCT and  COUNT
    how many custmoers order between '2017-02-01' AND '2017-02-0=28'
    SELET COUNT(DISTINCT customers_id) FROM orders
    WHERE order_time BETWEEN '2017-02-01' AND '2017-02-0=28'
```

```
SELECT COUNT(DISTINCT customer_id) FROM orders
WHERE order_time BETWEEN '2017-02-01' AND '2017-02-28';
```

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

COUNT(DISTINCT customer_id)
23

```
#### 4i. LIMIT
    allows to specify a number - EG : How many books do u want to select
    PRINT 5 RECENTLY REALEASED BOOKS.
    SELECT title,released_year FROM books ORDER BY 2 DESC LIMIT 5;

    print first 5-15 customers
    SELECT * FROM cutomers
    LIMIT 10 OFFSET 5
```

```
238
239 •   SELECT * FROM customers
240         ORDER BY last_name
241         LIMIT 10;
```

Result Grid | Filter Rows: | Edit:

id	first_name	last_name	gender	phone_number
10	Charlie	NULL	F	01139287883
22	Jennifer	NULL	F	NULL
6	Katie	Armstrong	F	01145787353
7	Michael	Bluth	M	01980289282
9	Buster	Bluth	M	01173456782
11	Lindsay	Bluth	F	01176923804
16	Gob	Bluth	M	01176985498
17	George	Bluth	M	01176984303

customers 105 ×

ex3

Exercise 3

1. From the customers table, select distinct last names and order alphabetically from A-Z.

2. From the orders table, select the first 3 orders placed by customer with id 1, in February 2017.

3. From the products table, select the name, price and coffee origin but rename the price to retail_price in the results set.

```sql
SELECT DISTINCT last_name FROM customers
ORDER BY last_name;
```

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

last_name
NULL
Armstrong
Bluth
Edmonds
Evans
Johnson
Law
Martin

```sql
SELECT *  FROM orders
WHERE order_time BETWEEN '2017-02-01' AND '2017-02-28'
AND customer_id =1
ORDER BY order_time ASC  -- no need to mention as its by def ASC.
LIMIT 3;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell C

id	product_id	customer_id	order_time
56	1	1	2017-02-01 08:10:14
64	1	1	2017-02-07 08:45:10
72	1	1	2017-02-13 08:37:45
NULL	NULL	NULL	NULL

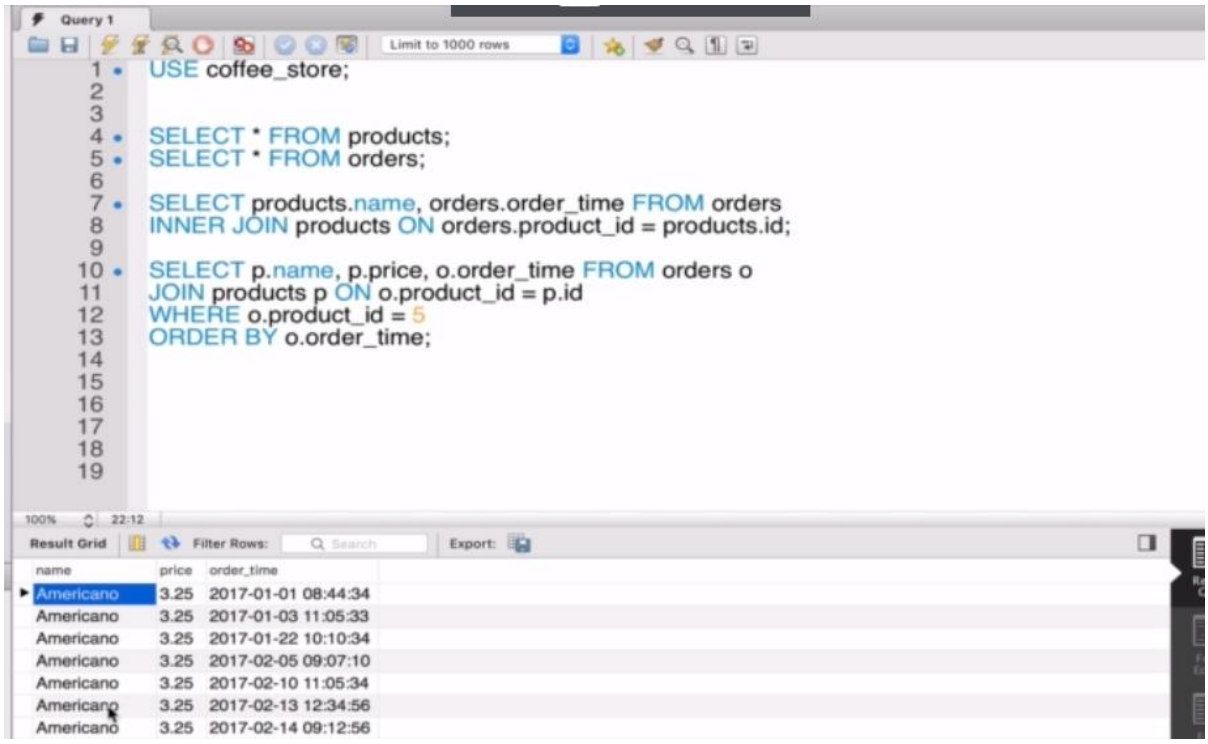
```sql
SELECT name , price AS retail_price FROM products;
```

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

name	retail_price
Espresso	3.50
Espresso	3.50
BLack	1.50
latte	1.50
machi	3.50
crude	4.50

JOINS
INNER JOIN or JOIN is same
RIGHT JOIN
LEFT JOIN
In MySQL , no such thing as FULL JOIN.

INNER



LEFT

ex in JOIN

tables given are

orders

Result Grid | Filter Rows: | Edit: | Expo

id	product_id	customer_id	order_time
1	1	1	2017-01-01 08:02:11
2	1	2	2017-01-01 08:05:16
3	5	12	2017-01-01 08:44:34
4	3	4	2017-01-01 09:20:02
5	1	9	2017-01-01 11:51:56
6	6	22	2017-01-01 13:07:10
7	1	1	2017-01-02 08:03:41
8	3	10	2017-01-02 09:15:22

customers

Result Grid | Filter Rows: | Edit: | Export/Import:

id	first_name	last_name	gender	phone_number
1	Chris	Martin	M	01123147789
2	Emma	Law	F	01123439899
3	Mark	Watkins	M	01174592013
4	Daniel	Williams	M	NULL
5	Sarah	Taylor	F	01176348290
6	Katie	Armstrong	F	01145787353
7	Michael	Bluth	M	01980289282
8	Kat	Nash	F	01176987789

customers 161 ×

products

255 SELECT * FROM products;

Result Grid | Filter Rows: | Edit: | Export/I

id	name	price
1	Espresso	3.50
2	Espresso	3.50
3	BLack	1.50
4	latte	1.50
5	machi	3.50
6	crude	4.50
7	Espresso	3.50
8	BLack	1.50

products 162 ×

1.

```
USE coffee_store;

-- 1. Select the order id and customers phone number for all orders of product id 4.
```

```
SELECT * FROM customers;
SELECT o.id , c.phone_number FROM orders o
JOIN  customers c  ON o.customer_id = c.id
WHERE o.product_id=4;
```

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

id	phone_number
15	01174592013
20	NULL
25	01176348290
29	01966947113
30	01176987789
32	01176984116

2.

```
-- 2. Select product name and order time for filter coffees sold between January 15th 2017
-- and February 14th 2017.
```

```
SELECT p.name, o.order_time FROM products p
JOIN orders o ON o.product_id=p.id
WHERE p.name='Espresso'
AND o.order_time BETWEEN '2017-01-15' AND '2017-02-14';
```

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

name	order_time
Espresso	2017-01-16 10:02:11
Espresso	2017-01-17 09:50:05
Espresso	2017-01-18 08:22:55
Espresso	2017-01-19 11:33:00
Espresso	2017-01-24 08:01:27
Espresso	2017-01-25 08:05:13
Espresso	2017-01-27 09:23:57
Espresso	2017-01-27 10:08:16

3.

```
-- 3. Select the product name and price and order time for all orders from females in January 2017.
```

1st line is very easy-

2nd line – Q says to choose prod for all orders -> JOIN orders o ON p.id = o.product_id (vice versa will produce diff o/p)

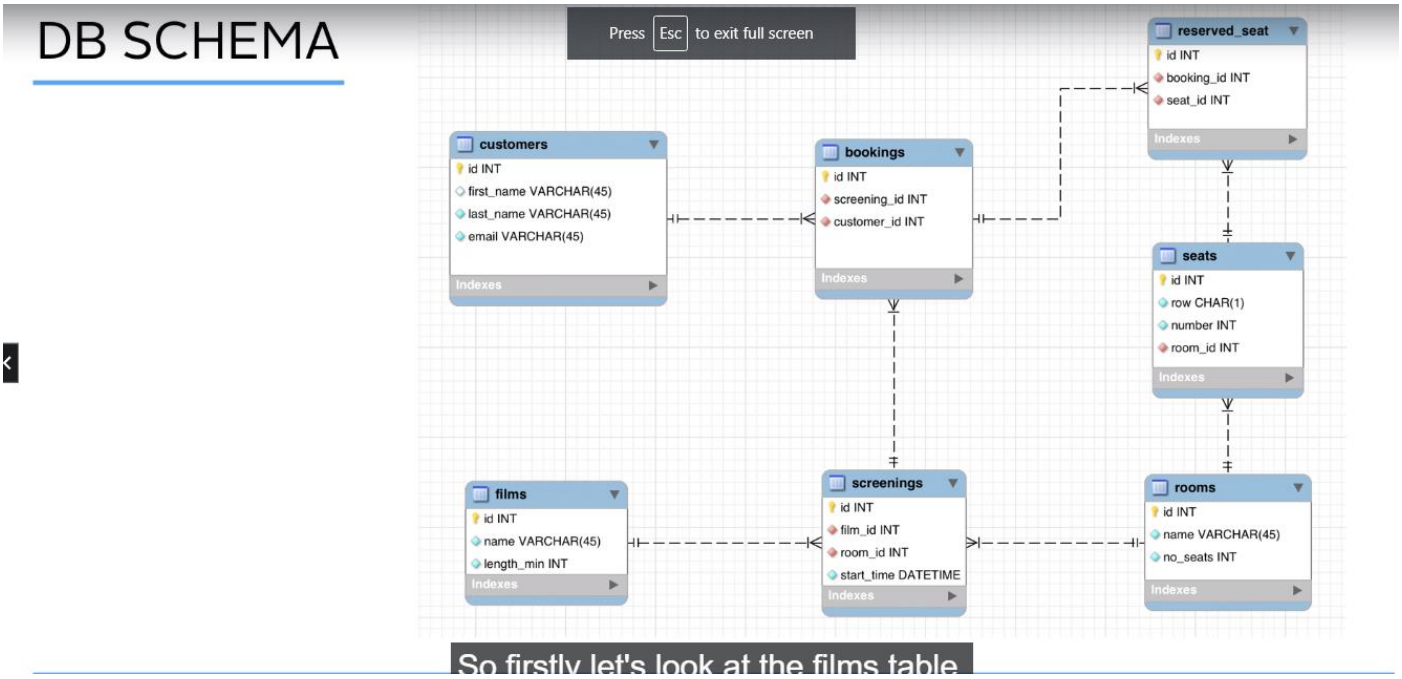
3rd- 1 says order for females – now JOIN order to customers not vv - JOIN customers c ON o.customer_id = c.id

```sql
SELECT p.name, p.price, o.order_time FROM products p
JOIN orders o ON p.id = o.product_id
JOIN customers c ON o.customer_id = c.id
WHERE c.gender='F'
AND  o.order_time BETWEEN '2017-01-01' AND '2017-01-31';
```

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

name	price	order_time
Espresso	3.50	2017-01-01 08:05:16
crude	4.50	2017-01-01 13:07:10
BLack	1.50	2017-01-02 09:15:22
Espresso	3.50	2017-01-02 10:10:10
machi	3.50	2017-01-03 11:05:33
BLack	1.50	2017-01-03 12:02:14
BLack	1.50	2017-01-04 11:23:43

m



films table

```
CREATE DATABASE cinema_booking_systm;
USE cinema_booking_systm;
-- films table
CREATE TABLE films(
 id INT PRIMARY KEY AUTO_INCREMENT,
 name VARCHAR(45) NOT NULL UNIQUE,
 length_min INT NOT NULL
);
SHOW TABLES;
SELECT * FROM films;
DESCRIBE films;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(45)	NO	UNI	NULL	
length_min	int	NO		NULL	

customers

```
CREATE TABLE customers(
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(45),
    last_name VARCHAR(45) NOT NULL,
    email VARCHAR(45) NOT NULL UNIQUE
);
SHOW TABLES;
DESCRIBE customers;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
first_name	varchar(45)	YES		NULL	
last_name	varchar(45)	NO		NULL	
email	varchar(45)	NO	UNI	NULL	

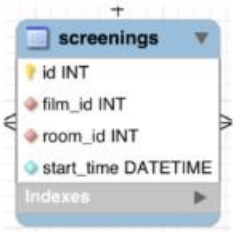
rooms

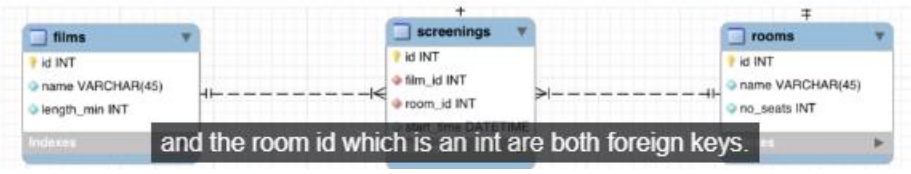
```
CREATE TABLE rooms(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(45) NOT NULL,
    no_seats INT NOT NULL
);
SHOW TABLES;
DESCRIBE rooms;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(45)	NO		NULL	
no_seats	int	NO		NULL	

screenings






```
CREATE TABLE screenings(
    id INT PRIMARY KEY AUTO_INCREMENT,
    film_id INT NOT NULL,
    room_id INT NOT NULL,
    start_time DATETIME NOT NULL,
    FOREIGN KEY  (film_id) REFERENCES films(id),
    FOREIGN KEY (room_id) REFERENCES rooms(id)
);
SHOW TABLES;
DESCRIBE screenings;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
film_id	int	NO	MUL	NULL	
room_id	int	NO	MUL	NULL	
start_time	datetime	NO		NULL	

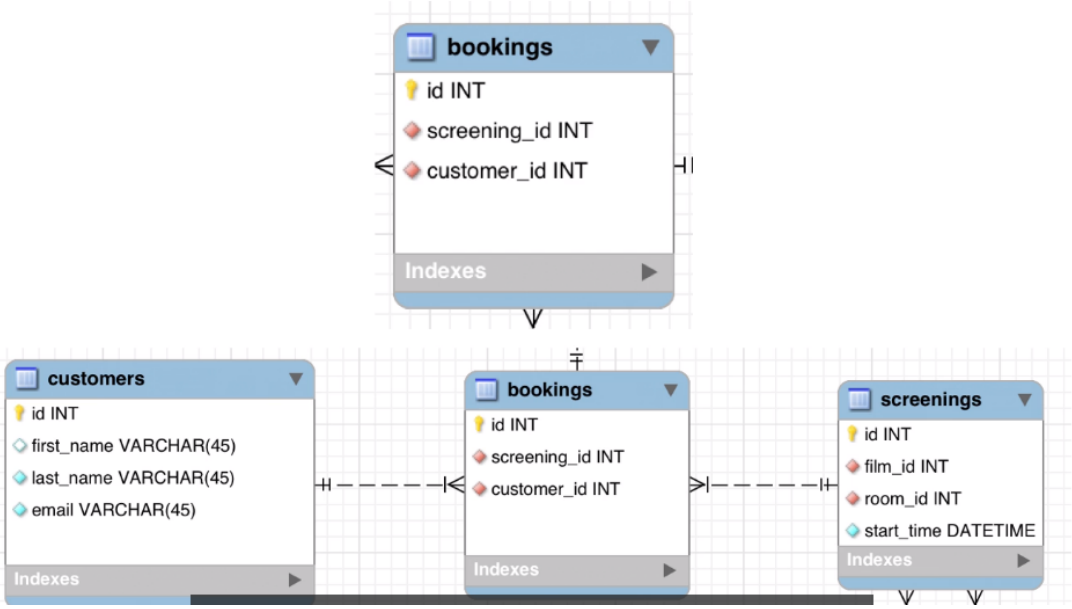
Seats

```sql
CREATE TABLE seats(
    id INT PRIMARY KEY AUTO_INCREMENT,
    rowes CHAR(1) NOT NULL,
    number INT NOT NULL,
    room_id INT NOT NULL,
    FOREIGN KEY (room_id)  REFERENCES rooms(id)

);
SHOW TABLES;
DESCRIBE seats;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
rowes	char(1)	NO		NULL	
number	int	NO		NULL	
room_id	int	NO	MUL	NULL	

booking



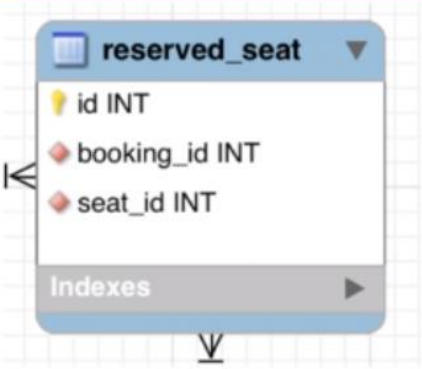
```sql
CREATE TABLE bookings(
    id INT PRIMARY KEY AUTO_INCREMENT,
    screening_id INT NOT NULL,
    customer_id INT NOT NULL,
    FOREIGN KEY (customer_id)  REFERENCES customers(id),
    FOREIGN KEY (screening_id)  REFERENCES screenings(id)

);
SHOW TABLES;
DESCRIBE bookings;
```

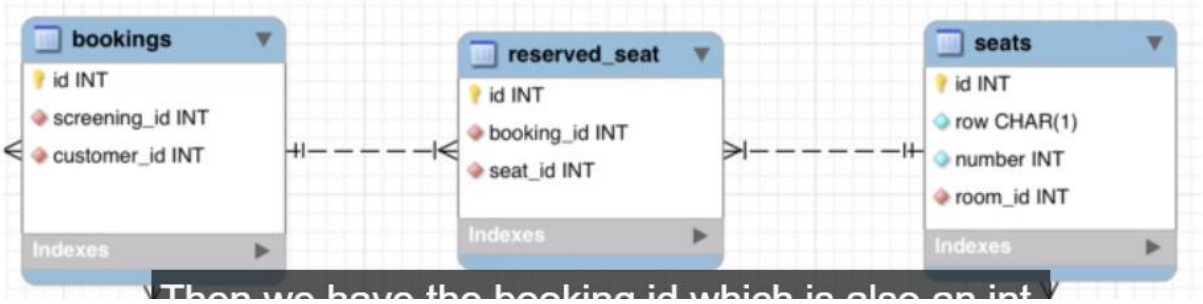
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
screening_id	int	NO	MUL	NULL	
customer_id	int	NO	MUL	NULL	

reserved_seat

one booking cust can reserved many seats





```sql
CREATE TABLE reserved_seat(
    id INT PRIMARY KEY AUTO_INCREMENT,
    booking_id INT NOT NULL,
    seat_id INT NOT NULL,
    FOREIGN KEY (booking_id)  REFERENCES bookings(id),
    FOREIGN KEY (seat_id)  REFERENCES seats(id)
);
SHOW TABLES;
DESCRIBE reserved_seat;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
booking_id	int	NO	MUL	NULL	
seat_id	int	NO	MUL	NULL	

SUBQUERIES

WHAT ARE SUBQUERIES

Subqueries are queries nested within other queries.

```sql
SELECT id, start_time FROM screenings
WHERE film_id IN
  (SELECT id FROM films
  WHERE length_min > 120)
;
```

```
### subqueries
    def -  queries nested within other queries - can be in WHERE caluse or FROM
            used in SELECT,INSERT,UPDATE or DELETE query
            NON-correlated and Correlated
```

NON-CORRELATED SUBQUERY

The inner query can run independently of the outer query.

```
SELECT id, start_time FROM screenings
WHERE film_id IN
  (SELECT id FROM films
  WHERE length_min > 120)
;
```

Inner query runs first and produces a result set, which is then used by the outer query.

which are then used by the outer query.

part1

```
USE cinema_booking_system;

SELECT id, start_time FROM screenings
WHERE film_id IN
  (SELECT id FROM films
  WHERE length_min > 120);

SELECT id FROM films
WHERE length_min > 120;


SELECT * FROM customers;
SELECT * FROM bookings;
```

id	start_time
10	2017-10-02 18:00:00
14	2017-10-02 19:30:00
15	2017-10-02 20:00:00
19	2017-10-03 19:00:00
23	2017-10-04 18:30:00
28	2017-10-04 19:30:00
31	2017-10-05 15:00:00
41	2017-10-07 13:30:00

we can see it's returned id and start time

b, customers that made a booking for screening_id =1

```
SELECT first_name, last_name, email FROM customers
WHERE id IN
(SELECT customer_id FROM bookings
WHERE screening_id = 1);
```

first_name	last_name	email
John	Smith	smithy@gmail.com
Cherry	Wang	cherrywa@gmail.com
Simon	Davis	sdavis@gmail.com

part2

no of seats reserved for booking_id

```
SELECT * FROM reserved_seat;

SELECT booking_id, COUNT(seat_id) FROM reserved_seat
GROUP BY booking_id;
```

booking_id	COUNT(seat_id)
1	3
2	2
3	2
4	2
5	1
6	2
7	2

find the max number of seats reserved by a part booking_id

since we are creating a direct table by

SELECT booking_id, COUNT(seat_id) AS no_seats FROM reserved_seat GROUP BY booking_id

SO, we have to provide a name too, say b, then just SELECT MAX(no_seats) FROM b

```
SELECT MAX(no_seats) FROM
(SELECT booking_id, COUNT(seat_id) AS no_seats FROM reserved_seat
GROUP BY booking_id) b;
```

MAX(no_seats)
6

WE can choose multiple columns from this direct table also

```
SELECT AVG(no_seats), MAX(no_seats) FROM
(SELECT booking_id, COUNT(seat_id) AS no_seats FROM reserved_seat
GROUP BY booking_id) b;

SELECT booking_id, COUNT(seat_id) AS no_seats FROM reserved_seat
GROUP BY booking_id;
```

AVG(no_seats)	MAX(no_seats)
1.8122	6

CORRELATED SUBQUERY

The inner query can't run independently of the outer query.

```
SELECT SCREENING_ID, CUSTOMER_ID,
(SELECT COUNT(SEAT_ID)
FROM RESERVED_SEAT WHERE BOOKING_ID = B.ID)
FROM BOOKINGS B;
```

The inner query runs for every row in the outer query.

the inner query is running multiple times

```
8  SELECT screening_id, customer_id,
9  (SELECT COUNT(seat_id)
10 FROM reserved_seat WHERE booking_id = b.id)
11 FROM bookings b
12 ORDER BY screening_id;
```

00% 43:10

Result Grid | Filter Rows: Search | Export:

screening_id	customer_id	(SELECT COUNT(seat_id)
1	4	2
1	12	2
1	16	4
2	2	3
2	6	2
2	18	2
3	20	1
4	8	2
4	12	2
4	67	2

now, running the inner query will give error

```
SELECT COUNT(seat_id)
FROM reserved_seat WHERE booking_id = b.id;
```

as booking as b was declared in the outer query

192 13:59:06 SELECT COUNT(seat_id) FROM reserved_seat WHERE booking_id = b.id LIMIT 0,... Error Code: 1054. Unknown column 'b.id' in 'where clause'

EX. a. non correlated query

SELECT name, length_min FROM films

WHERE length_min >

(SELECT AVG(length_min) as average FROM films);

```
1  USE cinema_booking_system;
2
3  -- 1. Select the film name and length for all films with a length greater than the average film le
4
5  SELECT name, length_min FROM films
6  WHERE length_min >
7  (SELECT AVG(length_min) FROM films);
8
```

100% 36:7

Result Grid Filter Rows: Search Export:

name	length_min
Blade Runner 2049	153
Geostorm	121
Jigsaw	118
Murder on the Orient Express	135
Breathe	117
Blade Runner	127

verification that it is NCQ

```
SELECT AVG(length_min) FROM films;
```

100% 25:9

Result Grid Filter Rows: Search Export:

AVG(length_min)
115.0833

```
-- 2. Select the maximum number and the minimum number of screenings for a particular film.
```

b.

AGAIN Ncq

SELECT film_id, COUNT(id) FROM screenings

GROUP BY film_id

```
SELECT film_id, COUNT(id) FROM screenings
GROUP BY film_id;
```

100% 17:14

Result Grid Filter Rows: Search Export:

film_id	COUNT(id)
1	24
2	25
3	20
4	18
5	26
6	19
7	22
8	14
9	18
10	6
11	6
12	15

gives each film id has how many screenings

- ➔ **now just select max and min from this table**

```
SELECT MAX(id), MIN(id) FROM
(SELECT film_id, COUNT(id) AS id FROM screenings
GROUP BY film_id) a;
```

100% 17:14

Result Grid Filter Rows: Search Export:

MAX(id)	MIN(id)
26	6

c. -- 3. Select each film name and the number of screenings for that film.

SELECT name

(SELECT COUNT(id) FROM screenings

WHERE film_id=f.id)

FROM films f;

- ➔ **we could have use group by screening but films and screenings are two diff table so we cant**

```
SELECT name,
(SELECT COUNT(id) FROM screenings
WHERE film_id = f.id
)
FROM films f;
```

100% 1:26

Result Grid | Filter Rows: Search | Export:

name	(SELECT COUNT(id) FROM screenings
Blade Runner	6
Blade Runner 2049	24
Breathe	6
Dunkirk	25
Geostorm	20
Jigsaw	26
Murder on the Orient Express	14
Paddington 2	18
The Death of Stalin	19
The Lego Ninjago Movie	22
Thor: Ragnarok	18

STRING EXS

```
#### UPPER, LOWER
    Syntax- SELECT UPPER(col1) AS new_col FROM table;
    SELECT UPPER('Hello World');  -- HELLO WORLD
    SELECT LOWER('Hello World');    -- hello world
    SELECT CONCAT('MY FAVORITE BOOK IS ', UPPER(title)) FROM books;

### CONCAT , CONCAT+ALIAS, CONCAT_WS , CONCAT+SUBSTRING+ALIAS
    Syntax - CONCAT(column, anotherColumn)
    SELECT CONCAT(author_fname,'<can enter any string>', author_lname) FROM books;
    SELECT author_fname AS first, author_lname AS last, CONCAT (author_fname, author_lname
) AS fullname FROM books;
    SELECT CONCAT_WS (' - ', title, author_fname, author_lname) FROM books;     - evenly s
paced with a symbol
    SELECT
         CONCAT
            (
            SUBSTRING(title,1,10),
            '...'
        ) AS 'short title'
        FROM books;
    SUBSTRING('Hello World', 1, 4)  -  Hell
    SUBSTRING('Hello World', 7) -  World
    SUBSTRING('Hello World', -3)    -  rld

    SELECT name FROM films;
    SELECT SUBSTRING(name,1,3) AS short_name FROM films;
```

```
-- Concatenate the film names and length from the films table.
SELECT CONCAT(name,": ",length_min) AS film_info FROM films;
```

Result Grid | Filter Rows: Search | Export:

film_info
Blade Runner 2049: 153
Dunkirk: 106
Geostorm: 121
Thor: Ragnarok: 107
Jigsaw: 116
The Death of Stalin: 98
The Lego Ninjago Movie: 101
Murder on the Orient Express: 135
Paddington 2: 88
Breathe: 117
Blade Runner: 127
Victoria and Abdul: 112

2.

```
-- Extract the customers email from the 5th character onwards.
SELECT SUBSTRING(email,5) AS email_short FROM customers;
```

100% 41:9

Result Grid | Filter Rows: Search | Export:

email_short
nb@gmail.com
landy@gmail.com
nks@gmail.com
nter@gmail.com
d@gmail.com
mer@gmail.com
le101@gmail.com

3.

```
-- Select the customers first name in lower case and their last name in upper case
-- for each customer with a last name of 'Smith'.

SELECT LOWER(first_name) AS first_name, UPPER(last_name) AS last_name FROM customers
WHERE last_name = 'Smith';
```

100% 27:15

Result Grid | Filter Rows: Search | Export:

first_name	last_name
john	SMITH
mark	SMITH
NULL	SMITH
stan	SMITH
NULL	SMITH
winston	SMITH

4.

```
-- Select the last 3 letters of each film name from the films table.

SELECT SUBSTRING(name,-3) AS film_name FROM films;
```

100% 51:19

Result Grid | Filter Rows: Search | Export:

film_name
ner
049
the
irk
orm
saw
ess
n 2
lin
vie
rok
dul

5.

```
-- Concatenate the first three letters in the first_name and last_name columns together
-- from the customers table.

SELECT CONCAT(SUBSTRING(first_name,1,3)," ",SUBSTRING(last_name,1,3)) AS short_name
FROM customers;
```

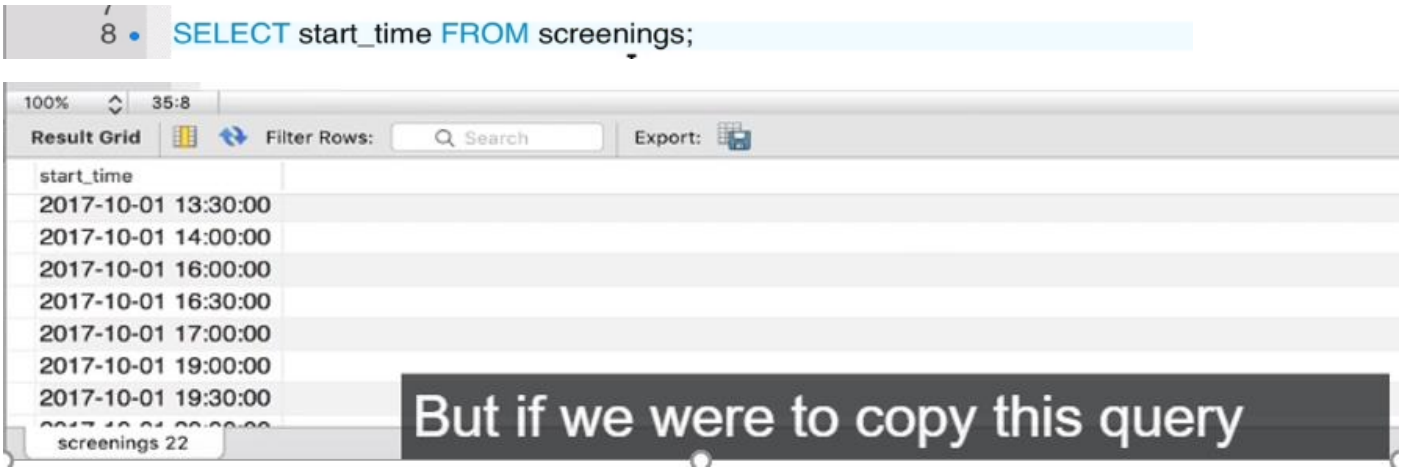
Result Grid | Filter Rows: Search | Export:

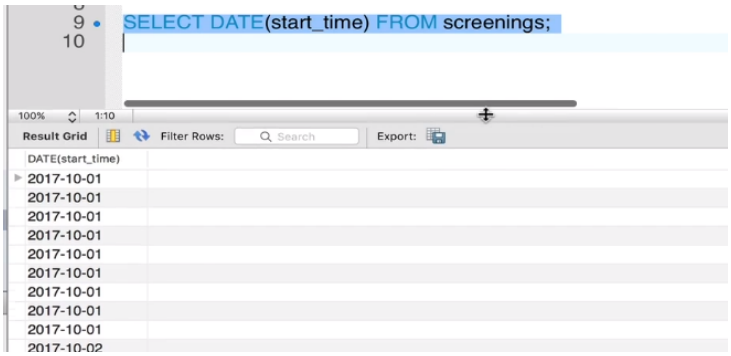
short_name
The Dav
Jer Mar
Joh Smi
Mar Wat
Emm Wat
Jav Pas
Cha Har
Mar Smi

DATE FUCTIONS

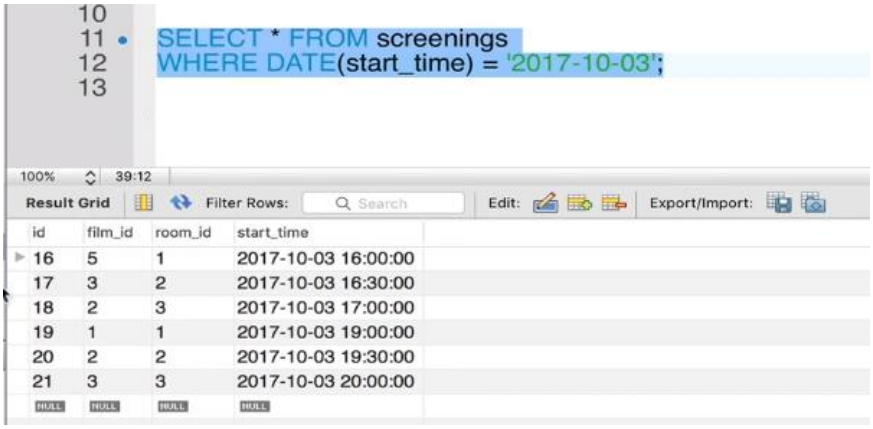
```
#### MYSQL FUNCTIONS
SELECT * FROM screenings
WHERE DATE(start_time) ='2017-10-03';

SELECT * FROM screenings
WHERE MONTH(start_time)='10';

SELECT * FROM screenings
WHERE YEAR(start_time) ='2017' - returns all data of 2017 year
```

data





adding WHERER



```
SELECT * FROM screenings
WHERE start_time = '2017-10-03';
```

100% 7:12

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	film_id	room_id	start_time
NULL	NULL	NULL	NULL

->

using BETWEEN AND DATE

```
SELECT * FROM screenings
WHERE DATE(start_time) BETWEEN '2017-10-03' AND '2017-10-05';
```

100% 1:16

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	film_id	room_id	start_time
16	5	1	2017-10-03 16:00:00
17	3	2	2017-10-03 16:30:00
18	2	3	2017-10-03 17:00:00
19	1	1	2017-10-03 19:00:00
20	2	2	2017-10-03 19:30:00
21	3	3	2017-10-03 20:00:00
22	3	1	2017-10-04 16:00:00
23	1	2	2017-10-04 16:30:00

MONTH FUNCTIONS

```
SELECT * FROM screenings
WHERE MONTH(start_time) = '10';
```

100% 32:13

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	film_id	room_id	start_time
1	2	1	2017-10-01 13:00:00
2	1	2	2017-10-01 13:30:00
3	2	3	2017-10-01 14:00:00
4	3	1	2017-10-01 16:00:00
5	6	2	2017-10-01 16:30:00
6	3	3	2017-10-01 17:00:00

YEAR

```
SELECT YEAR(start_time) FROM screenings;
```

100% 1:11

Result Grid | Filter Rows: Search | Export:

YEAR(start_time)
2017
2017
2017
2017
2017
2017
2017

EXs;

1.

```
-- Select the film id and start time from the screenings table for the date of 20th of October 2017.

SELECT film_id, start_time FROM screenings
WHERE DATE(start_time) = '2017-10-20';
```

100% 39:5

Result Grid | Filter Rows: Search | Export:

film_id	start_time
1	2017-10-20 16:00:00
1	2017-10-20 16:30:00
8	2017-10-20 17:00:00
2	2017-10-20 19:00:00
6	2017-10-20 19:30:00
4	2017-10-20 20:00:00

2.

```
-- Select all the data from the screenings table for the start time between the 6th and 13th of
-- October 2017.

SELECT * FROM screenings
WHERE DATE(start_time) BETWEEN '2017-10-06' AND '2017-10-13';
```

id	film_id	room_id	start_time
34	1	1	2017-10-06 16:00:00
35	5	2	2017-10-06 16:30:00
36	2	3	2017-10-06 17:00:00
37	4	1	2017-10-06 19:00:00
38	2	2	2017-10-06 19:30:00
39	4	3	2017-10-06 20:00:00
40	2	1	2017-10-07 13:00:00
41	1	2	2017-10-07 13:30:00
42	2	3	2017-10-07 14:00:00
81	7	3	2017-10-12 20:00:00
82	5	1	2017-10-13 16:00:00
83	2	2	2017-10-13 16:30:00
84	8	3	2017-10-13 17:00:00
85	9	1	2017-10-13 19:00:00
86	2	2	2017-10-13 19:30:00
87	9	3	2017-10-13 20:00:00

Query Stats

Execution Plan

3.

```
-- Select all the data from the screenings table for October 2017.

SELECT * FROM screenings
WHERE MONTH(start_time) = '10'
AND YEAR(start_time) = '2017';
```

100% 31:17

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	film_id	room_id	start_time
1	2	1	2017-10-01 13:00:00
2	1	2	2017-10-01 13:30:00
3	2	3	2017-10-01 14:00:00
4	3	1	2017-10-01 16:00:00
5	6	2	2017-10-01 16:30:00
6	3	3	2017-10-01 17:00:00
7	4	1	2017-10-01 19:00:00
8	6	2	2017-10-01 19:30:00
9	4	3	2017-10-01 20:00:00

1. -- Which films are over 2 hours long?

SELECT * FROM films

WHRER length_min>120;

```
SELECT * FROM films
WHERE length_min > 120;
```

100% 24:8

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id	name	length_min
1	Blade Runner 2049	153
3	Geostorm	121
8	Murder on the Orient Express	135
11	Blade Runner	127
NULL	NULL	NULL

2. -- Which film had the most screenings in October 2017

```
SELECT f.name, COUNT(s.film_id) AS showings FROM screenings s
JOIN films f ON f.id = s.film_id
GROUP BY film_id
ORDER BY showings DESC
LIMIT 1;
```

100% 9:9

Result Grid | Filter Rows: Search | Export: | Fetch rows:

name	showings
Jigsaw	26

3. -- How many bookings did the film 'Jigsaw' have in October 2017

```
SELECT id FROM screenings
WHERE film_id = 5;
```

100% 1:7

Result Grid | Filter Rows: Search | Edit: | Export/Import:

id
16
27
35
56
72
82
93
113
123

So that's the inner part of my sub-query.

inner query

```
SELECT COUNT(*) AS no_bookings FROM bookings
WHERE screening_id IN
(SELECT id FROM screenings
WHERE film_id = 5);
```

100% 20:8

Result Grid | Filter Rows: Search | Export:

no_bookings
59

4.

SELECT c.first_name, c.last_name, COUNT(b.id) AS no_bookings FROM bookings b

JOIN customers c ON c.id=b.customer_id

GROUP BY c.first_name, c.last_name

ORDER BY no_bookings DESC

LIMIT 5;

```
-- Which 5 customers made the most bookings in October 2017


SELECT c.first_name, c.last_name, COUNT(b.id) AS no_bookings FROM bookings b
JOIN customers c ON c.id = b.customer_id
GROUP BY c.first_name, c.last_name
ORDER BY no_bookings DESC
LIMIT 5;
```

100% 9:11

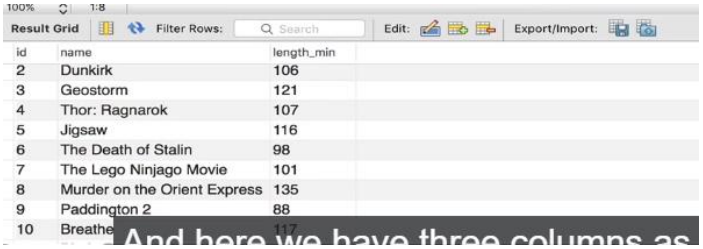
Result Grid | Filter Rows: Search | Export: | Fetch rows:

first_name	last_name	no_bookings
David	May	16
John	Smith	13
Ray	Wilkinson	12
Steve	Shoogan	12
David	Paul	12

5.

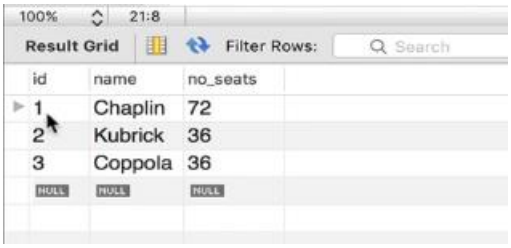
Which film was shown in the Chaplin room most often in October 2017

films



id	name	length_min
2	Dunkirk	106
3	Geostorm	121
4	Thor: Ragnarok	107
5	Jigsaw	116
6	The Death of Stalin	98
7	The Lego Ninjago Movie	101
8	Murder on the Orient Express	135
9	Paddington 2	88
10	Breathe	117

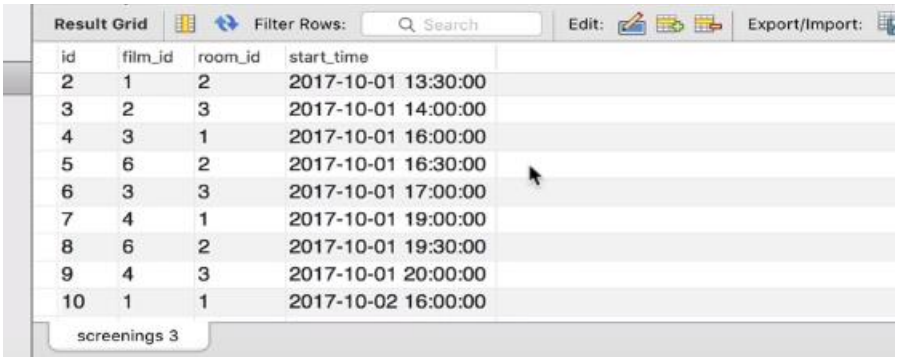
rooms



id	name	no_seats
1	Chaplin	72
2	Kubrick	36
3	Coppola	36
NULL	NULL	NULL

since rooms and films has no FK relation -> how to join now? lets see screening

screenings



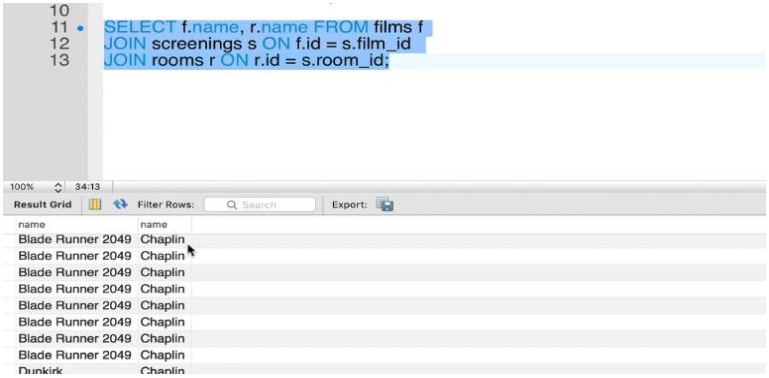
id	film_id	room_id	start_time
2	1	2	2017-10-01 13:30:00
3	2	3	2017-10-01 14:00:00
4	3	1	2017-10-01 16:00:00
5	6	2	2017-10-01 16:30:00
6	3	3	2017-10-01 17:00:00
7	4	1	2017-10-01 19:00:00
8	6	2	2017-10-01 19:30:00
9	4	3	2017-10-01 20:00:00
10	1	1	2017-10-02 16:00:00

screenings 3

now it has FK film_id ref to films table and room_id ref rooms table

- ➔ **have to join three tables**

so ,



```
SELECT f.name, r.name FROM films f
JOIN screenings s ON f.id = s.film_id
JOIN rooms r ON r.id = s.room_id;
```

name	name
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Dunkirk	Chaplin

we only interests in chaplin which as id=1.

```
SELECT f.name, r.name FROM films f
JOIN screenings s ON f.id = s.film_id
JOIN rooms r ON r.id = s.room_id
WHERE r.id = 1;
```

name	name
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin
Blade Runner 2049	Chaplin

- **have to group by film name and present the count of screening for each film,**
- **giving above result a name as no_screenings – want the max one – order by desc and limit 1;**

```
SELECT f.name, COUNT(r.name) AS no_screenings FROM films f
JOIN screenings s ON f.id = s.film_id
JOIN rooms r ON r.id = s.room_id
WHERE r.id = 1
GROUP BY f.name
ORDER BY no_screenings DESC
LIMIT 1;
```

name	no_screenings
Geostorm	13

-

6.

How many of the customers made a booking in October 2017

bookings

id	screening_id	customer_id
2	1	4
3	2	6
4	4	8
5	6	10

you can see we have an id colu

```
SELECT COUNT(DISTINCT(customer_id)) AS no_of_customers FROM bookings;
```