MYSQL2 -QUERIES

STEP 1: CREATE ANOTHER TABLE called shop in database1 AND INSERT VALUES into Shop table

mployeetable | database3 - Schema | employeetable | shop - Table | database1 - Schema

Limit to 1000 rows

```
INSERT INTO database1.shop  VALUES
    (1,'Z',3.45),(1,'Y',3.99),(2,'Z',10.99),(3,'Y',1.45),
    (3,'W',1.69),(3,'X',1.25),(4,'X',19.95);
```

STEP 2: list the contents of Shop table

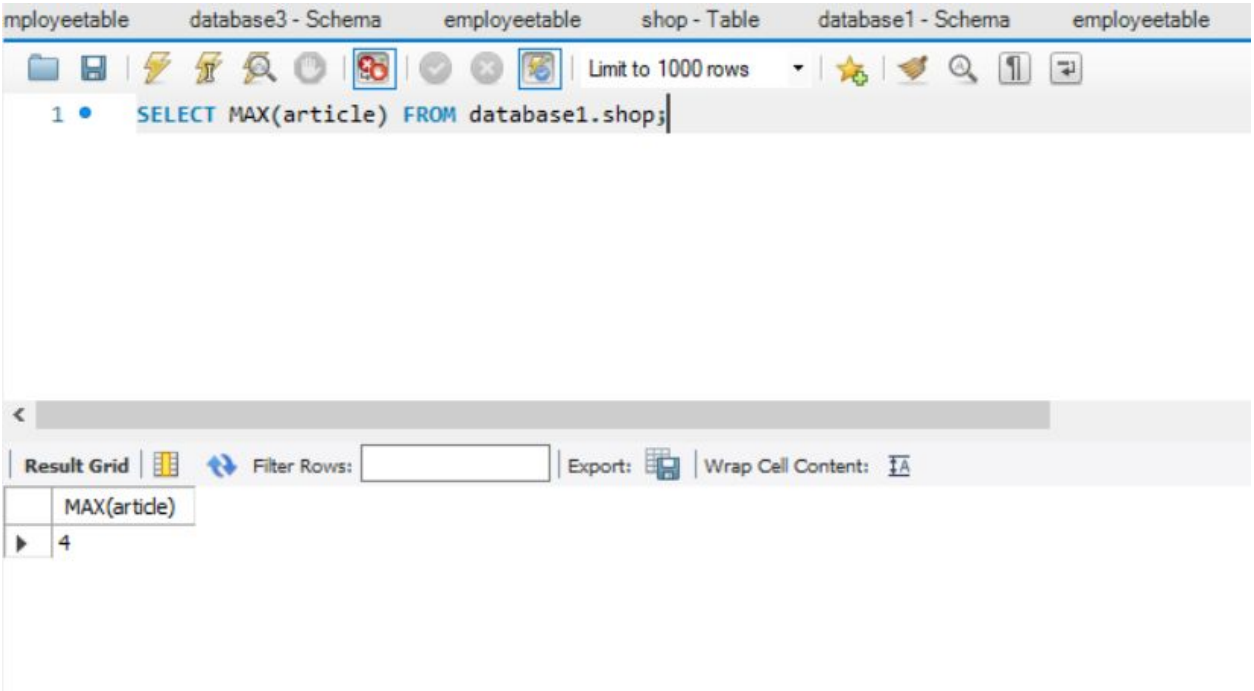
Limit to 1000 rows

```
SELECT * FROM database1.shop;
```

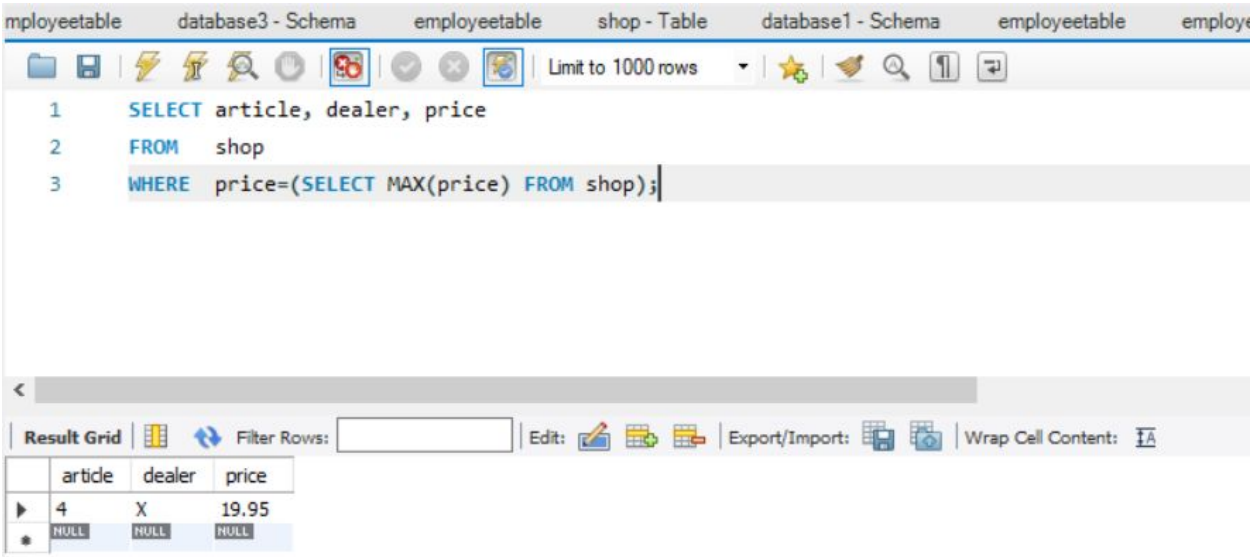
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

article	dealer	price
1	Y	3.99
1	Z	3.45
2	Z	10.99
3	W	1.69
3	X	1.25
3	Y	1.45
4	X	19.95
NULL	NULL	NULL

STEP 3: SELECT MAX(ARTICLE) from the list



STEP 4: Find the number, dealer and the price of the most expensive article in shop table



STEP 5: Find the number, dealer and the price of the most expensive article using LEFT JOIN and LIMIT CLAUSE in shop table.

Limit to 1000 rows

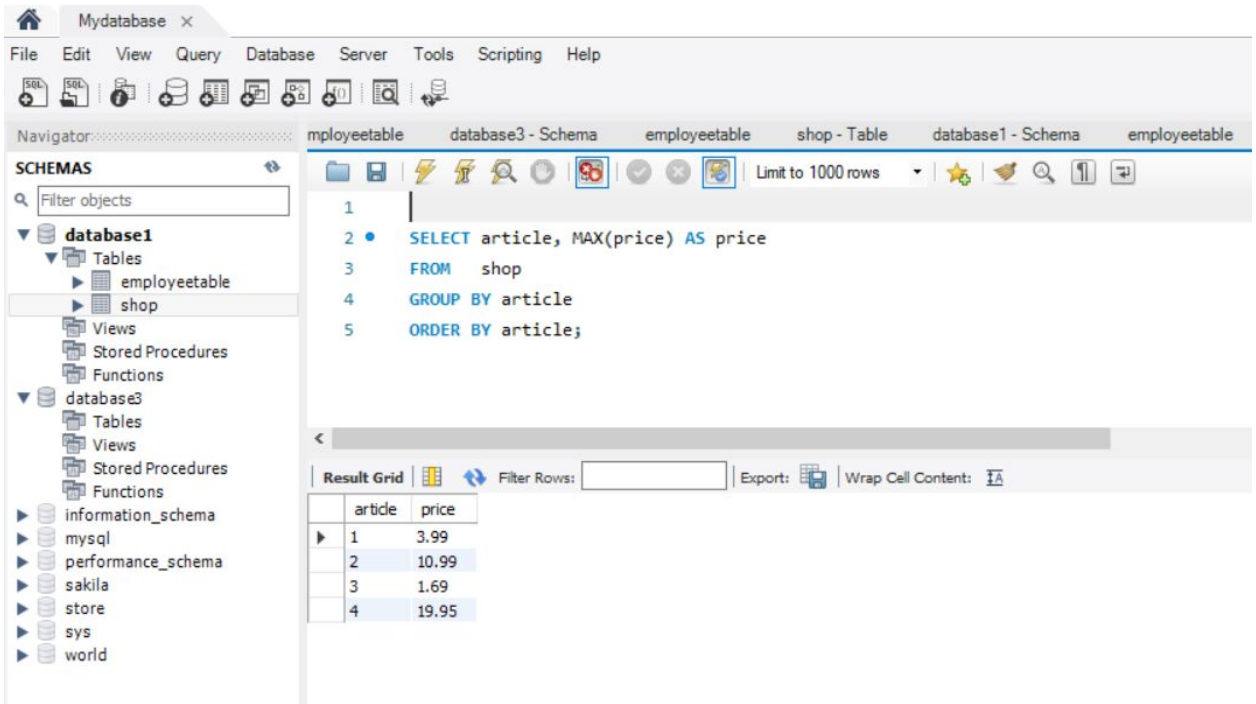
```
LEFT JOIN shop s2 ON s1.price < s2.price
WHERE s2.article IS NULL;

SELECT article, dealer, price
FROM shop
ORDER BY price DESC
LIMIT 1;
```

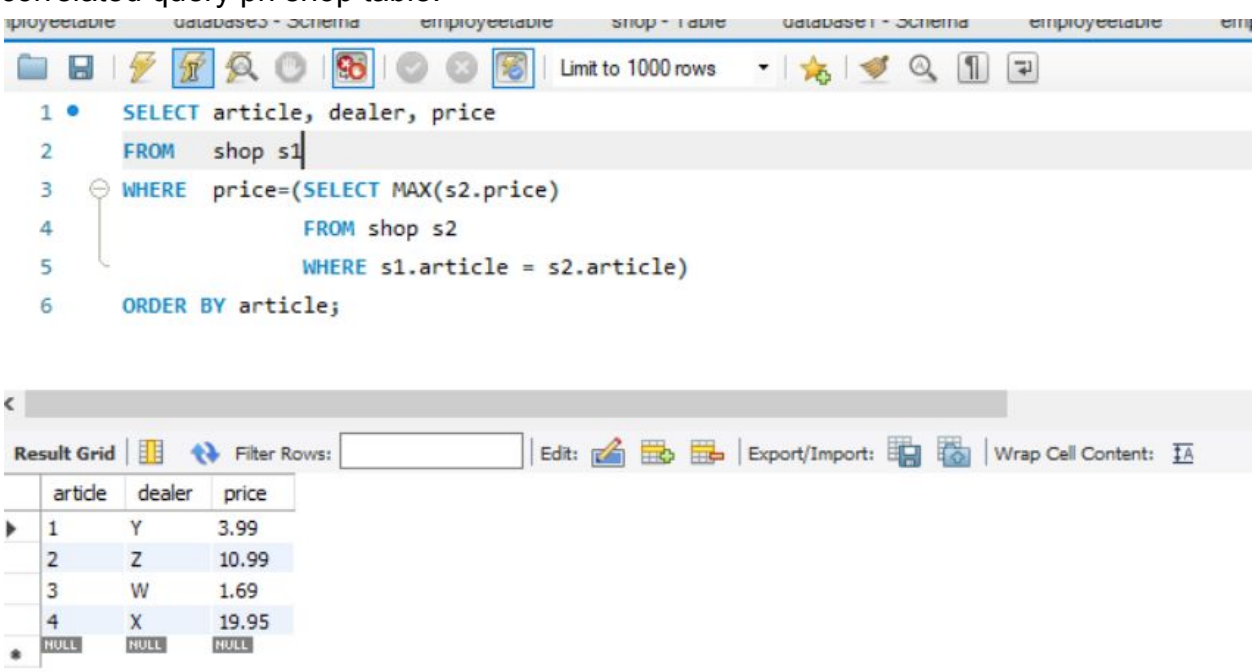
Result Grid | Filter Rows: | Edit:

article	dealer	price
4	X	19.95
NULL	NULL	NULL

STEP 6: Find the maximum of column per group in shop table.



STEP 7: For each article find the dealer or dealers with most expensive price using correlated query pn shop table.



STEP 8: For each article find the dealer or dealers with most expensive price using FROM clause using an un-correlated query. When there is no s2 price with a greater value LEFT JOIN works on the basis that s1 price is at its maximum.

```
SELECT s1.article, dealer, s1.price
FROM database1.shop s1
JOIN (
  SELECT article, MAX(price) AS price
  FROM database1.shop
  GROUP BY article) AS s2
  ON s1.article = s2.article AND s1.price = s2.price
ORDER BY article;
```

Result Grid

article	dealer	price
1	Y	3.99
2	Z	10.99
3	W	1.69
4	X	19.95

STEP 9: For each article find the dealer or dealers with most expensive price using LEFTJOIN clause using an un-correlated query in shop table.

```
SELECT s1.article, s1.dealer, s1.price
FROM database1.shop s1
LEFT JOIN database1.shop s2 ON s1.article = s2.article AND s1.price < s2.price
WHERE s2.article IS NULL
ORDER BY s1.article;
```

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

article	dealer	price
1	Y	3.99
2	Z	10.99
3	W	1.69
4	X	19.95

STEP 10: Query database1.shop table with user-defined queries- find min and max price from shop table

Query 1 | shop

Limit to 1000 rows

```
SELECT @min_price:=MIN(price),@max_price:=MAX(price) FROM database1.shop;
```

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

@min_price:=MIN(price)	@max_price:=MAX(price)
1.25	19.95

STEP 11: create two tables celebrity and shirt and insert values

Query 1 shop

Limit to 1000 rows

```sql
CREATE TABLE database1.celebrity (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE database1.shirt (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   shirtstyle ENUM('t-shirt', 'polo', 'dress') NOT NULL,
   shirtcolor ENUM('brown', 'lavender', 'orange', 'white', 'black') NOT NULL,
   owner SMALLINT UNSIGNED NOT NULL REFERENCES celebrity(id),
   PRIMARY KEY (id)
);
```

Query 1 shop

Limit to 1000 rows

```sql

INSERT INTO database1.celebrity VALUES (NULL, 'Ajith Kumar');

SELECT @last := LAST_INSERT_ID();

INSERT INTO database1.shirt VALUES
(NULL, 'polo', 'lavender', @last),
(NULL, 'dress', 'white', @last),
(NULL, 't-shirt', 'lavender', @last);

INSERT INTO database1.celebrity VALUES (NULL, 'Surya');

SELECT @last := LAST_INSERT_ID();
```

```
INSERT INTO database1.shirt VALUES
(NULL, 'dress', 'orange', @last),
(NULL, 'polo', 'brown', @last),
(NULL, 'dress', 'lavender', @last),
(NULL, 't-shirt', 'white', @last);
```

STEP 12: Check whether the table has been updated with contents

Query 1 shop

Limit to 1000 rows

```
(NULL, 'dress', 'lavender', @last),
(NULL, 't-shirt', 'white', @last);

SELECT * FROM database1.celebrity;
SELECT * FROM database1.shirt;
```

Result Grid Filter Rows: Edit: Export/Impo

id	shirtstyle	shirtcolor	owner
1	polo	lavender	1
2	dress	white	1
3	t-shirt	lavender	1
4	dress	orange	2
5	polo	brown	2
6	dress	lavender	2
7	t-shirt	white	2
NULL	NULL	NULL	NULL

STEP 13: Query using INNER JOIN and list shirts for celebrity Ajith other than color white..

Query 1 shop

Limit to 1000 rows

```
SELECT s.* FROM database1.celebrity c INNER JOIN database1.shirt s
   ON s.owner = c.id
 WHERE c.name LIKE 'Ajith Kumar%'
   AND s.shirtcolor <> 'white';
```

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

id	shirtstyle	shirtcolor	owner
1	polo	lavender	1
3	t-shirt	lavender	1

STEP 14: Query using INNER JOIN and list shirts for celebrity Surya other than color white..

Query 1 shop

Limit to 1000 rows

```
SELECT s.* FROM database1.celebrity c INNER JOIN database1.shirt s
   ON s.owner = c.id
 WHERE c.name LIKE 'Surya%'
   AND s.shirtcolor <> 'white';
```

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

id	shirtstyle	shirtcolor	owner
4	dress	orange	2
5	polo	brown	2
6	dress	lavender	2

STEP 15: Create Table visitsperday, Insert values and Query table using Bit group function to calculate the number of visits on a webpage eliminating duplicate entries.

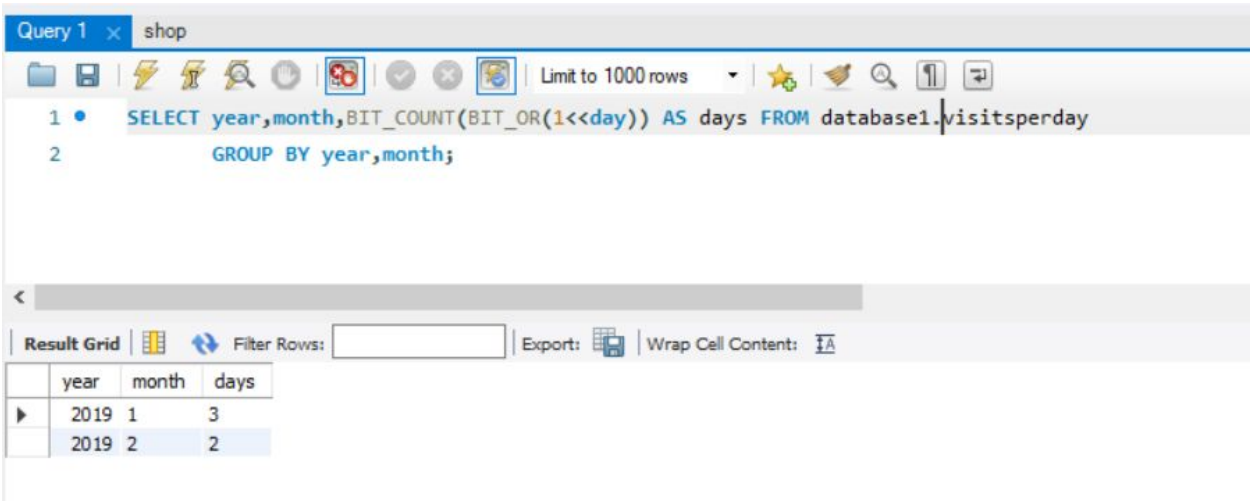
```
CREATE TABLE database1.visitsperday (year YEAR, month INT UNSIGNED,
                day INT UNSIGNED);
INSERT INTO database1.visitsperday VALUES(2019,1,1),(2019,1,20),(2019,1,30),(2019,2,2),
               (2019,2,23),(2019,2,23);
```

STEP 16: Query Table visitsperday

```
SELECT * FROM database1.visitsperday;
```

year	month	day
2019	1	1
2019	1	20
2019	1	30
2019	2	2
2019	2	23
2019	2	23

STEP 17: Query table using Bit group function to calculate the number of visits on a webpage eliminating duplicate entries.



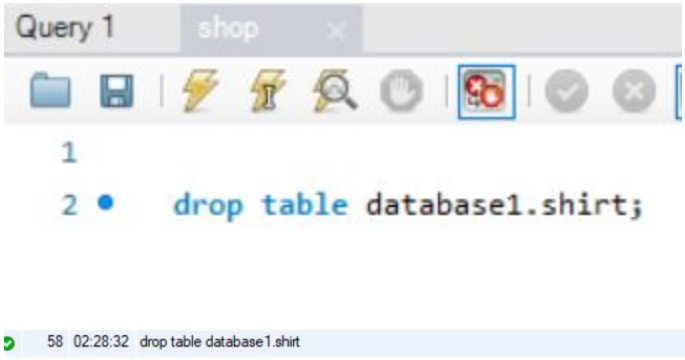
STEP 18 : DROP celebrity table.

```
drop table database1.celebrity;
```

on Output

Time	Action	Message
02:18:58	SELECT @last := LAST_INSERT_ID() LIMIT 0, 1000	1 row(s) returned
02:18:58	SELECT * FROM database1.shirt LIMIT 0, 1000	24 row(s) returned
02:21:50	drop table database1.celebrity	0 row(s) affected

STEP 19 : DROP shirt table.



STEP 20 : VERIFY DROP table database1 shirt table Query again in order to verify if the tables were dropped.