Sprint 3 – Mysql queries

Wenqian LU

The SQL script/s with query results (1st task)

Table: line_items

- First using **schema imarket_sql**

```
USE imarket_sql;
```

1. Select the entire line_item table.

```
SELECT * FROM line_item;
```

1.1. Select only the first 10 rows from the line_item table

```
SELECT * FROM line_item LIMIT 10;
```

1.2. Select only the columns sku, unit_price and date from the line_item table (and only the first 10 rows)

```
SELECT sku,unit_price,date FROM line_item LIMIT 10;
```

2. Count the total number of rows of the line_item table

```
SELECT COUNT(*) FROM line_item;
```

2.1. Count the total number of unique "sku" from the line_item table

```
SELECT COUNT(DISTINCT(sku)) FROM line_item;
```

3. Generate a table with the average price of each sku

```
SELECT sku, AVG(unit_price) FROM line_item  GROUP BY sku ;
```

3.1. ...now name the column of the previous query with the average price "avg_price", and sort the list that you by that column (bigger to smaller price)

```
SELECT sku, ROUND(AVG(unit_price),2) as Avg_price
FROM line_item
GROUP BY sku
ORDER BY Avg_price DESC;
```

4. Which products were bought in largest quantities? Select the “stock keeping unit” (sku) and product_quantity of the 100 products with the biggest "product quantity"

```
SELECT sku,product_quantity
FROM line_item
ORDER BY product_quantity DESC
LIMIT 100;
```

Table: orders

5. How many orders were placed in total?

```
SELECT COUNT(DISCOUNT id_order) FROM orders;
```

6. Make a count of orders by their state:

```
SELECT state, COUNT(DISTINCT id_order) FROM orders GROUP BY state;
```

7. Select all the orders placed in January of 2017

```
SELECT * FROM orders WHERE created_date LIKE '2017-01-%';
```

8. Count the number of orders of your previous select query (i.e. How many orders were placed in January of 2017?)

```
SELECT COUNT(DISTINCT id_order) FROM orders WHERE created_date LIKE '2017-01-%';
```

9. How many orders were cancelled on 2017?

```
SELECT COUNT(id_order) FROM orders
WHERE state = 'cancelled' AND  YEAR(created_date) ='2017';
```

10. How many orders have been placed each month of the year?

```
SELECT MONTH(created_date), COUNT(DISTINCT id_order)
FROM orders
GROUP BY  Month(created_date);
```

11. What is the total amount paid in all the orders?

```
SELECT SUM(total_paid) FROM orders;
```

12. What is the average amount paid per order?

```
SELECT AVG(total_paid) AS 'paid per order' FROM orders;
```

12.1 Give a result to the previous question with only 2 decimals

```
SELECT ROUND AVG(total_paid),2) AS 'paid per order' FROM orders;
```

13. What is the date of the newest order? And the oldest?

```
SELECT MAX(created_date) FROM orders;
SELECT MIN(created_date) FROM orders;
```

-- What is the day with the highest amount of completed orders (and how many completed orders were placed that day)?

```
SELECT id_order, created_date,state
FROM orders
WHERE state ='completed' ORDER BY id_order DESC LIMIT 1;
```

-- What is the day with the highest amount paid (and how much was paid that day)?

```
SELECT total_paid, created_date,state
FROM
WHERE state ='completed'
ORDER BY total_paid DESC LIMIT 1;
```

Table.products

-- How many products are there?

```
SELECT COUNT(*) FROM products;
```

-- How many brands?

```
SELECT COUNT(distinct brand) FROM products;
```

-- How many categories?

```
SELECT COUNT(distinct manual_categories) FROM products;
```

-- How many products per brand & products per category?

```
SELECT brand,manual_categories ,count(*) AS 'product per brand&procutcts'
FROM products  GROUP BY 1,2;
```

-- What's the average price per brand and the average price per category?

```
--SELECT brand, ROUND(avg(price),2) AS 'avg price per brand'
  FROM products  GROUP BY brand;
--SELECT manual_categories, ROUND(avg(price),2) AS 'avg price per category'
  FROM products
  GROUP BY manual_categories;
```

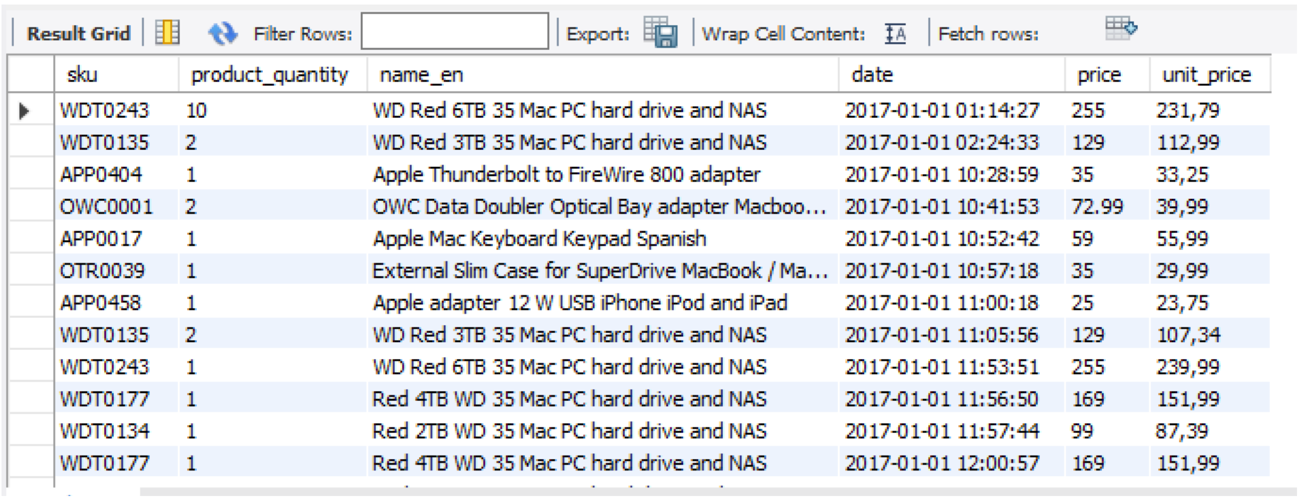
-- What's the name and description of the most expensive product per brand and per category?

```
SELECT name_en,short_desc_en,price,brand,manual_categories
 FROM products WHERE (price, brand,manual_categories) IN
(SELECT  max(price), brand,manual_categories
 FROM products  GROUP BY brand,manual_categories;
```

The SQL script for database creation and query results

- **Query 1.** Our first query should return the "sku", "product_quantity", "date" and "unit_price" from the line_item table together with the "name" and the "price" of each product from the "products" table. We want only products present in both tables.

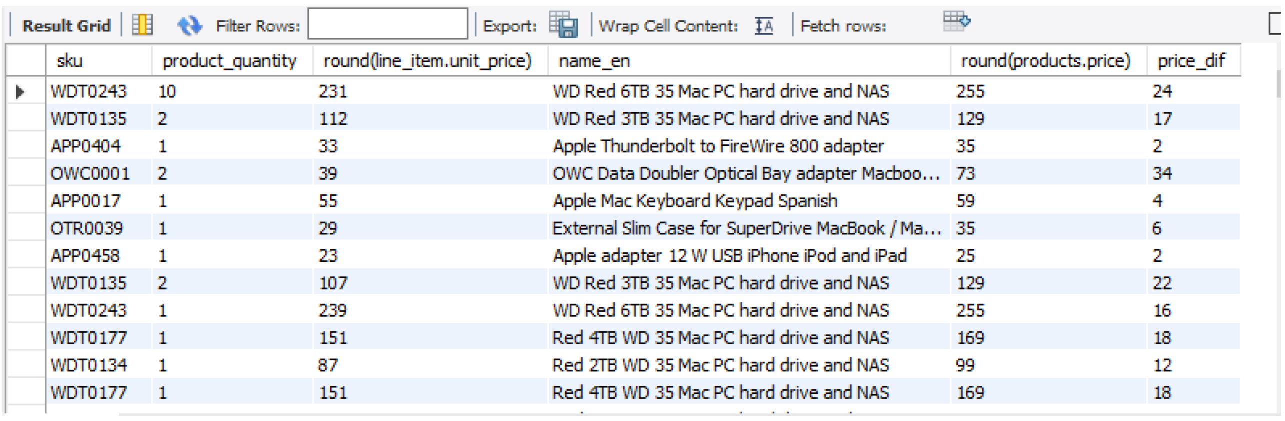
```
USE imarket_sql;

SELECT line_item.sku, line_item.product_quantity, products.name_en,

          line_item.date,products.price,line_item.unit_price

FROM line_item

 INNER JOIN products ON line_item.sku = products.sku;
```



sku	product_quantity	name_en	date	price	unit_price
WDT0243	10	WD Red 6TB 35 Mac PC hard drive and NAS	2017-01-01 01:14:27	255	231,79
WDT0135	2	WD Red 3TB 35 Mac PC hard drive and NAS	2017-01-01 02:24:33	129	112,99
APP0404	1	Apple Thunderbolt to FireWire 800 adapter	2017-01-01 10:28:59	35	33,25
OWC0001	2	OWC Data Doubler Optical Bay adapter Macboo...	2017-01-01 10:41:53	72.99	39,99
APP0017	1	Apple Mac Keyboard Keypad Spanish	2017-01-01 10:52:42	59	55,99
OTR0039	1	External Slim Case for SuperDrive MacBook / Ma...	2017-01-01 10:57:18	35	29,99
APP0458	1	Apple adapter 12 W USB iPhone iPod and iPad	2017-01-01 11:00:18	25	23,75
WDT0135	2	WD Red 3TB 35 Mac PC hard drive and NAS	2017-01-01 11:05:56	129	107,34
WDT0243	1	WD Red 6TB 35 Mac PC hard drive and NAS	2017-01-01 11:53:51	255	239,99
WDT0177	1	Red 4TB WD 35 Mac PC hard drive and NAS	2017-01-01 11:56:50	169	151,99
WDT0134	1	Red 2TB WD 35 Mac PC hard drive and NAS	2017-01-01 11:57:44	99	87,39
WDT0177	1	Red 4TB WD 35 Mac PC hard drive and NAS	2017-01-01 12:00:57	169	151,99

- **Query 2.** You might notice that the unit_price from the line_item table and the price from the product table is not the same. Let's investigate that! Extend your previous query by adding a column with the difference in price. Name that column price_difference.

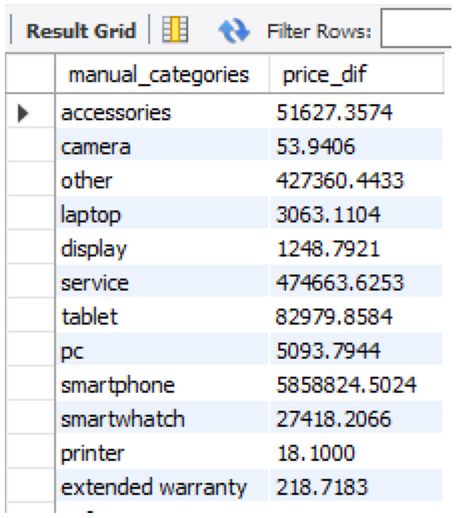
```
USE imarket_sql;

SELECT line_item.sku, line_item.product_quantity, round(line_item.unit_price),

                    products.name_en, round(products.price),

    round((products.price) - (line_item.unit_price)) as price_dif

FROM line_item

INNER JOIN products

ON line_item.sku = products.sku;
```



sku	product_quantity	round(line_item.unit_price)	name_en	round(products.price)	price_dif
WDT0243	10	231	WD Red 6TB 35 Mac PC hard drive and NAS	255	24
WDT0135	2	112	WD Red 3TB 35 Mac PC hard drive and NAS	129	17
APP0404	1	33	Apple Thunderbolt to FireWire 800 adapter	35	2
OWC0001	2	39	OWC Data Doubler Optical Bay adapter Macboo...	73	34
APP0017	1	55	Apple Mac Keyboard Keypad Spanish	59	4
OTR0039	1	29	External Slim Case for SuperDrive MacBook / Ma...	35	6
APP0458	1	23	Apple adapter 12 W USB iPhone iPod and iPad	25	2
WDT0135	2	107	WD Red 3TB 35 Mac PC hard drive and NAS	129	22
WDT0243	1	239	WD Red 6TB 35 Mac PC hard drive and NAS	255	16
WDT0177	1	151	Red 4TB WD 35 Mac PC hard drive and NAS	169	18
WDT0134	1	87	Red 2TB WD 35 Mac PC hard drive and NAS	99	12
WDT0177	1	151	Red 4TB WD 35 Mac PC hard drive and NAS	169	18

- **Query 3.** Build a query that outputs the price difference that you just calculated, grouping products by category. Round the result.

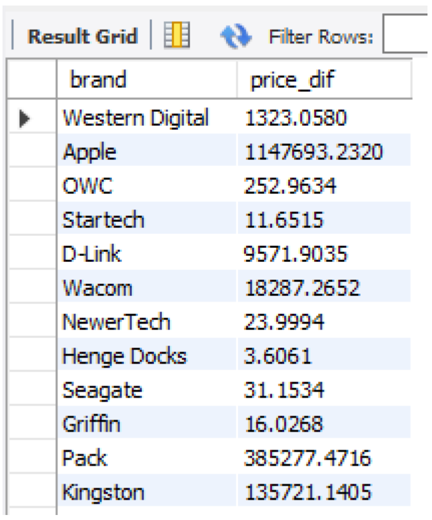
```
USE imarket_sql;

SELECT products.manual_categories,

    AVG(round((products.price) -
(line_item.unit_price))) as price_dif

FROM line_item

INNER JOIN products

ON line_item.sku = products.sku

GROUP BY manual_categories;
```



manual_categories	price_dif
accessories	51627.3574
camera	53.9406
other	427360.4433
laptop	3063.1104
display	1248.7921
service	474663.6253
tablet	82979.8584
pc	5093.7944
smartphone	5858824.5024
smartwhatch	27418.2066
printer	18.1000
extended warranty	218.7183

- **Query 4.** Create the same query as before (calculating the price difference between the line_item and the products tables, but now grouping by brands instead of categories.

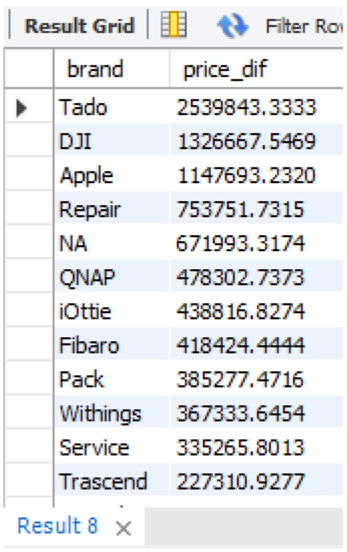
```
USE imarket_sql;

SELECT products.brand,

    AVG(round((products.price) - (line_item.unit_price))) as
price_dif

FROM line_item

INNER JOIN products

ON line_item.sku = products.sku

GROUP BY brand;
```



brand	price_dif
Western Digital	1323.0580
Apple	1147693.2320
OWC	252.9634
Startech	11.6515
D-Link	9571.9035
Wacom	18287.2652
NewerTech	23.9994
Henge Docks	3.6061
Seagate	31.1534
Griffin	16.0268
Pack	385277.4716
Kingston	135721.1405

- **Query 5.** Let's focus on the brands with a big price difference: run the same query as before, but now limiting the results to only brands with an avg_price_dif of more than 50000. Order the results by avg_price_dif (bigger to smaller).

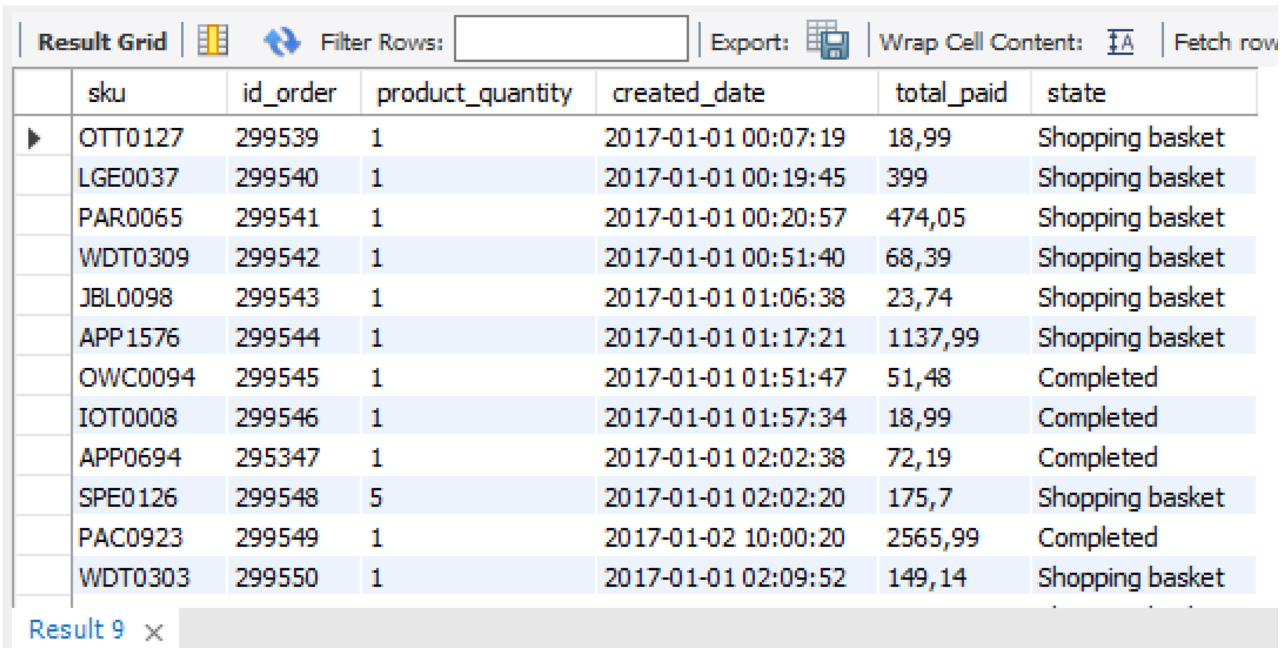
```
USE imarket_sql;

SELECT products.brand,

    AVG(round((products.price) - (line_item.unit_price))) as price_dif

FROM line_item

INNER JOIN products

ON line_item.sku = products.sku

GROUP BY brand

HAVING price_dif > 50000

ORDER BY price_dif DESC;
```



brand	price_dif
Tado	2539843.3333
DJI	1326667.5469
Apple	1147693.2320
Repair	753751.7315
NA	671993.3174
QNAP	478302.7373
iOttie	438816.8274
Fibaro	418424.4444
Pack	385277.4716
Withings	367333.6454
Service	335265.8013
Trascend	227310.9277

- **Query 6.** First, we will connect each product (sku) from the line_item table to the orders table. We only want sku that have been in any order. This table will contain duplicates, and we're ok with that. We will group and count this information later.

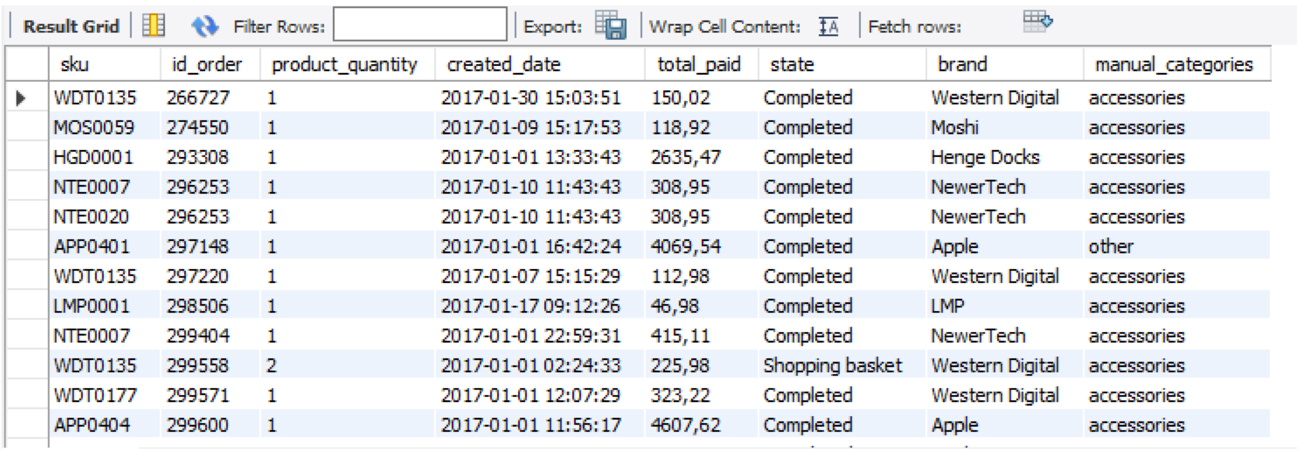
```
USE imarket_sql;

SELECT line_item.sku,
line_item.id_order,line_item.product_quantity,orders.created_date,orders.total_paid,orders.state

FROM line_item

JOIN orders

ON line_item.id_order = orders.id_order;
```



sku	id_order	product_quantity	created_date	total_paid	state
OTT0127	299539	1	2017-01-01 00:07:19	18,99	Shopping basket
LGE0037	299540	1	2017-01-01 00:19:45	399	Shopping basket
PAR0065	299541	1	2017-01-01 00:20:57	474,05	Shopping basket
WDT0309	299542	1	2017-01-01 00:51:40	68,39	Shopping basket
JBL0098	299543	1	2017-01-01 01:06:38	23,74	Shopping basket
APP1576	299544	1	2017-01-01 01:17:21	1137,99	Shopping basket
OWC0094	299545	1	2017-01-01 01:51:47	51,48	Completed
IOT0008	299546	1	2017-01-01 01:57:34	18,99	Completed
APP0694	295347	1	2017-01-01 02:02:38	72,19	Completed
SPE0126	299548	5	2017-01-01 02:02:20	175,7	Shopping basket
PAC0923	299549	1	2017-01-02 10:00:20	2565,99	Completed
WDT0303	299550	1	2017-01-01 02:09:52	149,14	Shopping basket

- **Query 7.** Now, add to the previous query the brand and the category from the products table to this query.

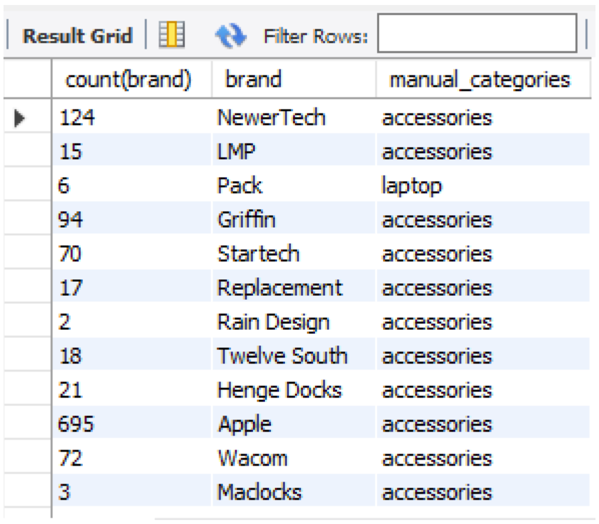
```
USE imarket_sql;

SELECT line_item.sku,
line_item.id_order,line_item.product_quantity,orders.created_date,orders.total_paid,

orders.state,products.brand,products.manual_categories

FROM line_item

INNER JOIN products

ON products.sku=line_item.sku

INNER JOIN orders

ON line_item.id_order = orders.id_order;
```



sku	id_order	product_quantity	created_date	total_paid	state	brand	manual_categories
WDT0135	266727	1	2017-01-30 15:03:51	150,02	Completed	Western Digital	accessories
MOS0059	274550	1	2017-01-09 15:17:53	118,92	Completed	Moshi	accessories
HGD0001	293308	1	2017-01-01 13:33:43	2635,47	Completed	Henge Docks	accessories
NTE0007	296253	1	2017-01-10 11:43:43	308,95	Completed	NewerTech	accessories
NTE0020	296253	1	2017-01-10 11:43:43	308,95	Completed	NewerTech	accessories
APP0401	297148	1	2017-01-01 16:42:24	4069,54	Completed	Apple	other
WDT0135	297220	1	2017-01-07 15:15:29	112,98	Completed	Western Digital	accessories
LMP0001	298506	1	2017-01-17 09:12:26	46,98	Completed	LMP	accessories
NTE0007	299404	1	2017-01-01 22:59:31	415,11	Completed	NewerTech	accessories
WDT0135	299558	2	2017-01-01 02:24:33	225,98	Shopping basket	Western Digital	accessories
WDT0177	299571	1	2017-01-01 12:07:29	323,22	Completed	Western Digital	accessories
APP0404	299600	1	2017-01-01 11:56:17	4607,62	Completed	Apple	accessories

- **Query 8.** Let's keep working on the same query: now we want to keep only Cancelled orders. Modify this query to group the results from the previous query, first by category and then by brand, adding in both cases a count so we know which categories and which brands are most times present in Cancelled orders.

```
USE imarket_sql;

SELECT count(brand),brand,manual_categories FROM (

SELECT line_item.sku, line_item.id_order,orders.state,products.brand,products.manual_categories

FROM line_item

INNER JOIN products

ON products.sku=line_item.sku

INNER JOIN orders

ON line_item.id_order = orders.id_order

HAVING orders.state = 'cancelled' ) as new  GROUP BY  brand,manual_categories ;
```


Result Grid | Filter Rows:

count(brand)	brand	manual_categories
124	NewerTech	accessories
15	LMP	accessories
6	Pack	laptop
94	Griffin	accessories
70	Startech	accessories
17	Replacement	accessories
2	Rain Design	accessories
18	Twelve South	accessories
21	Henge Docks	accessories
695	Apple	accessories
72	Wacom	accessories
3	Maclocks	accessories

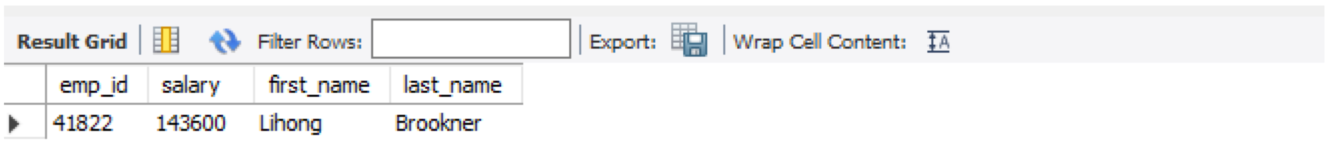
A brief report on the SQL database organisation.

Import database Organisation

Query 1

What is the employee id of the highest-paid employee?

```
USE organisation;

SELECT salaries.emp_id,salaries.salary,employees.first_name,employees.last_name

FROM salaries

INNER JOIN employees

ON employees.emp_id=salaries.emp_id

ORDER BY salary DESC LIMIT 1;
```

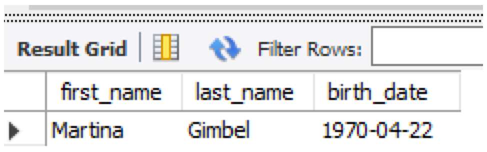


emp_id	salary	first_name	last_name
41822	143600	Lihong	Brookner

Query 2

What is the name of the youngest employee?

```
USE organisation;

SELECT first_name, last_name,birth_date

FROM employees

ORDER BY birth_date DESC LIMIT 1;
```



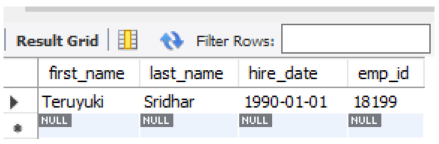
first_name	last_name	birth_date
Martina	Gimbel	1970-04-22

Query 3

What is the name of the first hired employee?

(*Hint*: use order by clause on 2 variables & employee with lowest employee id is the 1st employee)

```
USE organisation;

SELECT first_name, last_name,hire_date,emp_id

FROM employees

ORDER BY hire_date,emp_id  ASC LIMIT 1;
```

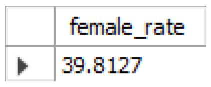


	first_name	last_name	hire_date	emp_id
▶	Teruyuki	Sridhar	1990-01-01	18199
*	NULL	NULL	NULL	NULL

Query 4

What percentage of employees are Female?

```
USE organisation;

SELECT (SELECT count(*) from employees

where gender='F') /

(SELECT count(*) from employees)  *100 as female_rate ;
```

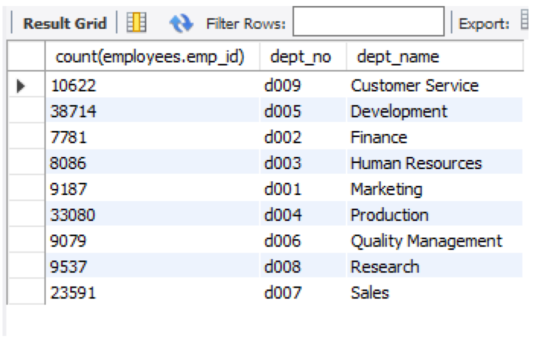


	female_rate
▶	39.8127

Query 5

Show the employee count by department name wise, sorted alphabetically on department name.

```
USE organisation;

SELECT
count(employees.emp_id),dept_emp.dept_no,departmen
ts.dept_name

FROM employees

INNER JOIN dept_emp

ON dept_emp.emp_id=employees.emp_id

INNER JOIN departments

ON departments.dept_no=dept_emp.dept_no

GROUP BY dept_name

ORDER BY dept_name ASC;
```

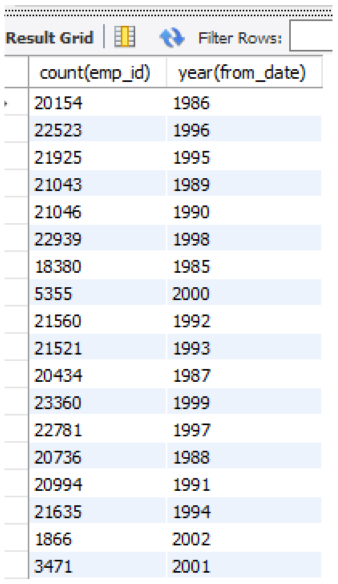


	count(employees.emp_id)	dept_no	dept_name
▶	10622	d009	Customer Service
	38714	d005	Development
	7781	d002	Finance
	8086	d003	Human Resources
	9187	d001	Marketing
	33080	d004	Production
	9079	d006	Quality Management
	9537	d008	Research
	23591	d007	Sales

Query 6

Count the number of employees by each calendar year (take the value of year from *from_date*)

```
USE organisation;

SELECT count(emp_id),Year(from_date)

FROM dept_emp

GROUP BY Year(from_date);
```



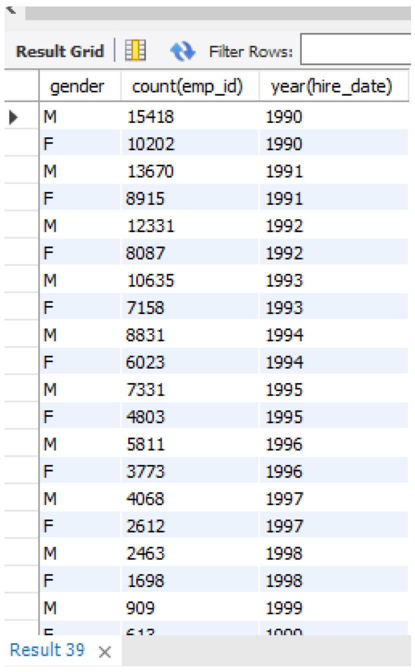
count(emp_id)	year(from_date)
20154	1986
22523	1996
21925	1995
21043	1989
21046	1990
22939	1998
18380	1985
5355	2000
21560	1992
21521	1993
20434	1987
23360	1999
22781	1997
20736	1988
20994	1991
21635	1994
1866	2002
3471	2001

Query 7

Count the number of employees by each calendar year (take the value of year from *from_date*) ordered by the calendar year excluding all years before 1990.

Divide the employee count based on gender.

```
USE organisation;

SELECT gender,count(emp_id),Year(hire_date)

FROM employees

WHERE (EXTRACT(Year from hire_date)) > 1989

GROUP BY Year(hire_date),gender

ORDER BY Year(hire_date) ASC;
```



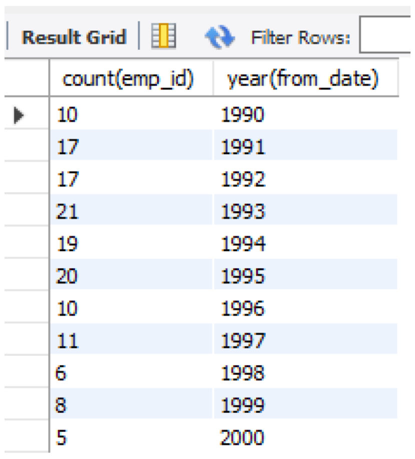
gender	count(emp_id)	year(hire_date)
M	15418	1990
F	10202	1990
M	13670	1991
F	8915	1991
M	12331	1992
F	8087	1992
M	10635	1993
F	7158	1993
M	8831	1994
F	6023	1994
M	7331	1995
F	4803	1995
M	5811	1996
F	3773	1996
M	4068	1997
F	2612	1997
M	2463	1998
F	1698	1998
M	909	1999
F	613	1999

Query 8

What is the number of managers hired each calendar year? The table should look like below:

(*Hint*: The manager's details are stored in *dept_manager* table)

```
USE organisation;

SELECT count(emp_id),Year(from_date)

FROM dept_manager

GROUP BY Year(from_date)

ORDER BY Year(from_date) ASC;
```

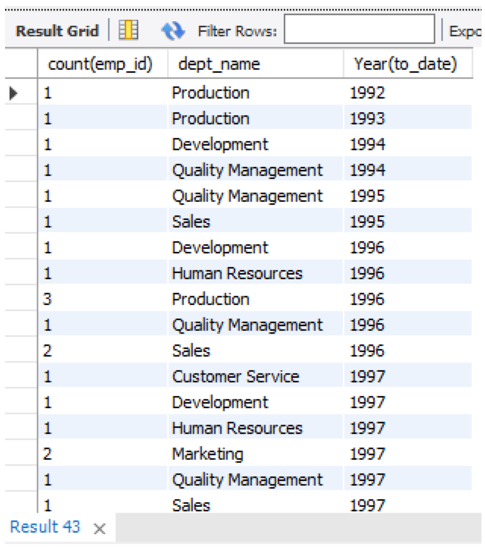


count(emp_id)	year(from_date)
10	1990
17	1991
17	1992
21	1993
19	1994
20	1995
10	1996
11	1997
6	1998
8	1999
5	2000

Query 9

What will be the department-wise break up of managers?

```
USE organisation;

SELECT count(emp_id),dept_name ,Year(to_date)

FROM (SELECT
dept_manager.emp_id,dept_manager.to_date,departments.dept_
name

FROM dept_manager

INNER JOIN departments

ON departments.dept_no=dept_manager.dept_no) as new

GROUP BY Year(to_date),dept_name

ORDER BY Year(to_date) ASC;
```



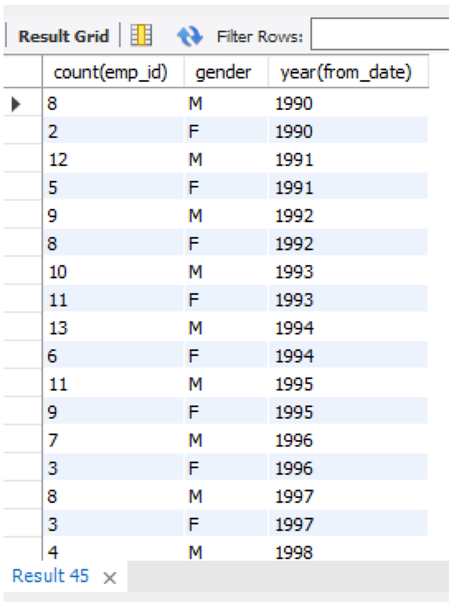
count(emp_id)	dept_name	Year(to_date)
1	Production	1992
1	Production	1993
1	Development	1994
1	Quality Management	1994
1	Quality Management	1995
1	Sales	1995
1	Development	1996
1	Human Resources	1996
3	Production	1996
1	Quality Management	1996
2	Sales	1996
1	Customer Service	1997
1	Development	1997
1	Human Resources	1997
2	Marketing	1997
1	Quality Management	1997
1	Sales	1997

Query 10

What is the number of male managers and female managers hired each calendar year from the year 1990 onwards?

(***sample** output*)

```
USE organisation;

SELECT count(emp_id),gender ,Year(from_date)

FROM (SELECT
dept_manager.emp_id,dept_manager.from_date,employees.gend
er

FROM dept_manager

INNER JOIN employees

ON departments.emp_id=employees.emp_id) as new

GROUP BY Year(from_date),gender

ORDER BY Year(from_date) ASC;
```



count(emp_id)	gender	year(from_date)
8	M	1990
2	F	1990
12	M	1991
5	F	1991
9	M	1992
8	F	1992
10	M	1993
11	F	1993
13	M	1994
6	F	1994
11	M	1995
9	F	1995
7	M	1996
3	F	1996
8	M	1997
3	F	1997
4	M	1998