

HOMWORK WEEK 1

(handout for students)

TASK 1

USE PARTS DB TO WRITE THE FOLLOWING QUERIES

1. Find the name of each part where the weight is more than 14.

ANSWER:

1 – I check all parts to see what weight each has:

USE parts;

SELECT * FROM part;

The screenshot shows a database query tool interface. At the top, a tab labeled 'Query 1' is active. Below the tab is a toolbar with various icons, including a 'Limit to 1000 rows' dropdown. The query text is as follows:

```
1 • USE parts;  
2 • SELECT * FROM part;  
3
```

Below the query editor, a 'Result Grid' is displayed. It has a toolbar with 'Filter Rows', 'Export', and 'Wrap Cell Content' options. The grid contains the following data:

	P_ID	PNAME	COLOUR	WEIGHT	CITY
▶	P1	NUT	RED	12	LONDON
	P2	BOLT	GREEN	17	PARIS
	P3	SCREW	BLUE	17	ROME
	P4	SCREW	RED	14	LONDON
	P5	CAM	BLUE	12	PARIS
	P6	COG	RED	19	LONDON

Below the result grid, a tab labeled 'part 1' is active. It shows an 'Output' section with a dropdown menu set to 'Action Output'. The output table is as follows:

	#	Time	Action	Message
✓	1	17:14:02	USE parts	0 row(s) affected
✓	2	17:14:02	SELECT * FROM part LIMIT 0, 1000	6 row(s) returned

2 – only 2 results have their weight below 14 and 1 is equal to 14, so 3 results should be excluded out of 6 parts total available in the part table – I run query to only show parts with weight above 14:

USE parts;
 SELECT * FROM part WHERE weight > 14;

Query 1

1 • USE parts;
 2 • SELECT * FROM part WHERE weight > 14;
 3

	P_ID	PNAME	COLOUR	WEIGHT	CITY
▶	P2	BOLT	GREEN	17	PARIS
	P3	SCREW	BLUE	17	ROME
	P6	COG	RED	19	LONDON

Output

Action Output

#	Time	Action	Message
✓ 1	17:14:02	USE parts	0 row(s) affected
✓ 2	17:14:02	SELECT * FROM part LIMIT 0, 1000	6 row(s) returned
✓ 3	17:17:53	USE parts	0 row(s) affected
✓ 4	17:17:53	SELECT * FROM part WHERE weight > 14 LIMIT 0, 1000	3 row(s) returned

3 – result is as expected and looks correct, 3 results have weight over 14.

2. Find all **unique** supplier(s) where their status is equal to 20.

ANSWER:

1 – first I check all suppliers, to see where is the data of all suppliers in parts DB and how it looks like:

SQL File 3* x

1 • USE parts;

2 • SELECT * FROM supplier;

3

<

Result Grid | Filter Rows:

	S_ID	SNAME	STATUS	CITY
▶	S1	SMITH	20	LONDON
	S2	JONES	10	PARIS
	S3	BLAKE	30	PARIS
	S4	CLARK	20	LONDON
	S5	ADAMS	30	ATHENS

supplier 1 x

2 – data has both the required information (supplier name and status), so I go on to performing requested query: select those with status being equal to 20:

USE parts;
 SELECT DISTINCT s.s_ID, s.sname, s.status FROM supplier AS s WHERE s.status = 20;

SQL File 3* x

1 • USE parts;

2 • SELECT DISTINCT s.s_id, s.sname, s.status FROM supplier AS s WHERE s.status = 20;

3

<

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

	s_id	sname	status
▶	S1	SMITH	20
	S4	CLARK	20

3 – result shows both unique suppliers with status = 20.

NOTE 1 : While in this particular case using the keyword “DISTINCT” or not using it makes no difference, it is a good practice and makes a clean, understandable code to include it.

NOTE 2 : I also used the “supplier AS s” here in hopes it will turn off the blue colour from “status” if it’s written as “s.status” but it’s still blue, as if it were a keyword instead of a column name. I’ve highlighted that this is a table’s column, not a keyword by adding the table’s name to this column’s name.

TASK 2

USE SHOP SALES DB TO WRITE THE FOLLOWING QUERIES

1. Find out how many sales (amount) were recorded each week, per day (where available)

○	This	would	look	like:
	Week	1,	Tuesday,	£x
	Week	1,	Wednesday,	£x
	Week	2,	Monday,	£x
	Week 2, Friday, £x			

ANSWER:

USE shop;

```
SELECT s.week, s.day, s.salesamount FROM sales1 AS s  
ORDER BY s.week, s.day ASC;
```

I’d need to add the word “Week” to each digit in “week” column to make it look like “Week 1 , Tuesday, GBP SalesAmount” and I was not exactly sure if that is what was asked of me in this exercise. It was my understanding that we are only to group and order the result, and sort it ascending.

SQL File 3*

Limit to 1000 rows

```

1 • USE shop;
2 • SELECT s.week, s.day, s.salesamount FROM sales1 AS s
3   ORDER BY s.week, s.day ASC;

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	week	day	salesamount
▶	1	Saturday	43.11
	1	Tuesday	44.27
	2	Monday	56.25
	3	Tuesday	9.99
	4	Monday	77.00
	4	Wednesday	86.81
	5	Monday	98.42
	5	Saturday	73.90
	5	Tuesday	74.32
	6	Friday	74.02

2. Change the name of salesperson Inga to be Annette instead, but only where Ignas Sales are <50.

First checking if there is an Inga in the database and how many records are there to know, so I can know if I will get the correct result – there is only 1 Inga sale and it is for under 10 pounds.

SQL File 3*

```

1 • USE shop;
2 • SELECT s.salesperson, s.salesamount FROM sales1 AS s;
3   -- UPDATE sales1 AS s
4   -- SET s.salesperson = 'Annette'
5   -- WHERE s.salesperson = 'Inga' AND s.salesamount > 50;

```

Result Grid

	salesperson	salesamount
▶	Frank	56.25
	Frank	74.32
	Bill	98.42
	Bill	73.90
	Josie	44.27
	Manfred	77.00
	Inga	9.99
	Manfred	86.81
	Josie	74.02
	Manfred	43.11

Then running the requested query – swapping “Inga” for “Annette” in records where Inga’s sale is under 50 pounds – this means in only 1 record, the only one of Inga’s:

```

USE shop;
UPDATE sales1 AS s
SET s.salesperson = 'Annette'
WHERE s.salesperson = 'Inga' AND s.salesamount < 50;
SELECT s.salesperson, s.salesamount FROM sales1 AS s;

```

Query 1

```

1 • USE shop;
2 • UPDATE sales1 AS s
3   SET s.salesperson = 'Annette'
4   WHERE s.salesperson = 'Inga' AND s.salesamount < 50;
5 • SELECT s.salesperson, s.salesamount FROM sales1 AS s;

```

Result Grid

	salesperson	salesamount
▶	Frank	56.25
	Frank	74.32
	Bill	98.42
	Bill	73.90
	Josie	44.27
	Manfred	77.00
	Annette	9.99
	Manfred	86.81
	Josie	74.02
	Manfred	43.11

It shows no Inga and the sales of 9.99 is ascribed to Annette so it looks like it worked 😊

3. Find out how many sales the London office logged
 - Note we're looking for quantity here - e.g. if London did 6 sales, then output would be 6)

Query 1

```

1 • USE shop;
2 • SELECT
3   COUNT(DISTINCT s.salesamount) FROM sales1 AS s
4   WHERE store = 'LONDON';

```

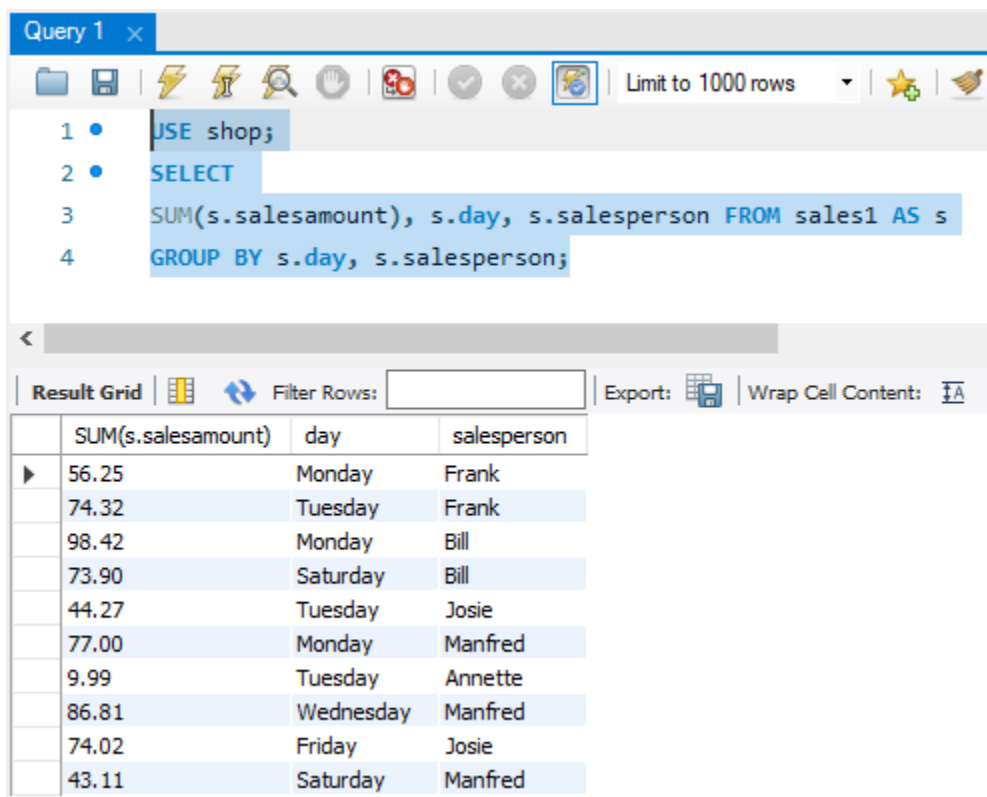
Result Grid

	COUNT(DISTINCT s.salesamount)
▶	6

```
USE shop;
SELECT
COUNT(DISTINCT s.salesamount) FROM sales1 AS s
WHERE store = 'LONDON';
```

4. Find the total (sum) sales amount by each person by day

```
USE shop;
SELECT
SUM(s.salesamount), s.day, s.salesperson FROM sales1 AS s
GROUP BY s.day, s.salesperson;
```



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • USE shop;
2 • SELECT
3   SUM(s.salesamount), s.day, s.salesperson FROM sales1 AS s
4   GROUP BY s.day, s.salesperson;
```

Below the query editor, the results are displayed in a table with the following columns: SUM(s.salesamount), day, and salesperson. The results are as follows:

SUM(s.salesamount)	day	salesperson
56.25	Monday	Frank
74.32	Tuesday	Frank
98.42	Monday	Bill
73.90	Saturday	Bill
44.27	Tuesday	Josie
77.00	Monday	Manfred
9.99	Tuesday	Annette
86.81	Wednesday	Manfred
74.02	Friday	Josie
43.11	Saturday	Manfred

5. How much (sum) each person sold for between week 1 and week 3

```
USE shop;
SELECT
SUM(s.salesamount), s.salesperson FROM sales1 AS s
WHERE s.week BETWEEN 1 AND 3
GROUP BY s.salesperson;
```


Query 1

```

1 • USE shop;
2 • SELECT
3   SUM(s.salesamount), s.salesperson FROM sales1 AS s
4   WHERE s.week BETWEEN 1 AND 3
5   GROUP BY s.salesperson;

```

Result Grid

	SUM(s.salesamount)	salesperson
▶	56.25	Frank
	44.27	Josie
	9.99	Annette
	43.11	Manfred

Checking:

Query 1

```

1 • USE shop;
2 • SELECT
3   SUM(s.salesamount), s.salesperson, s.week FROM sales1 AS s
4   WHERE s.week BETWEEN 1 AND 3
5   GROUP BY s.salesperson;

```

Result Grid

	SUM(s.salesamount)	salesperson	week
▶	56.25	Frank	2
	44.27	Josie	1
	9.99	Annette	3
	43.11	Manfred	1

These results are for weeks 1 till 3, correct.

Results from weeks 4 and 5 are omitted:

Query 1

```

1 • USE shop;
2 • SELECT
3   SUM(s.salesamount), s.salesperson, s.week FROM sales1 AS s
4   GROUP BY s.salesperson;

```

Result Grid

	SUM(s.salesamount)	salesperson	week
▶	130.57	Frank	2
	172.32	Bill	5
	118.29	Josie	1
	206.92	Manfred	4
	9.99	Annette	3

6. For each store:

- The total of their sales;
- The number of sales;
- Their average sales;
- Their lowest sales amount;
- Their highest sales amount.

Query 1

```

1 • USE shop;
2 • SELECT
3   SUM(s.salesamount), COUNT(s.salesamount), AVG(s.salesamount),
4   MIN(s.salesamount), MAX(s.salesamount), s.store FROM sales1 AS s
5   GROUP BY s.store;

```

Result Grid

	SUM(s.salesamount)	COUNT(s.salesamount)	AVG(s.salesamount)	MIN(s.salesamount)	MAX(s.salesamount)	store
▶	421.18	6	70.196667	44.27	98.42	London
	216.91	4	54.227500	9.99	86.81	Dusseldorf

```

USE shop;
SELECT
SUM(s.salesamount),          COUNT(s.salesamount),          AVG(s.salesamount),
MIN(s.salesamount), MAX(s.salesamount), s.store FROM sales1 AS s
GROUP BY s.store;

```

7. Find the average (AVG) monetary sales amount achieved by each store

The screenshot shows a SQL query editor with the following query:

```
1 • USE shop;  
2 • SELECT  
3 ✖ AVG(s.salesamount), s.store FROM sales1 AS s  
4 GROUP BY s.store;
```

The query is executed, and the result grid shows the following data:

AVG(s.salesamount)	store
70.196667	London
54.227500	Dusseldorf

USE shop;
SELECT
AVG(s.salesamount), s.store FROM sales1 AS s
GROUP BY s.store;

8. Count the number of sales by each person if they had less than 3 sales for the past period

The screenshot shows a SQL query editor with the following query:

```
1 • USE shop;  
2 • SELECT  
3 COUNT(s.salesamount), s.salesperson FROM sales1 AS s  
4 GROUP BY s.salesperson  
5 HAVING COUNT(s.salesamount) < 3;
```

The query is executed, and the result grid shows the following data:

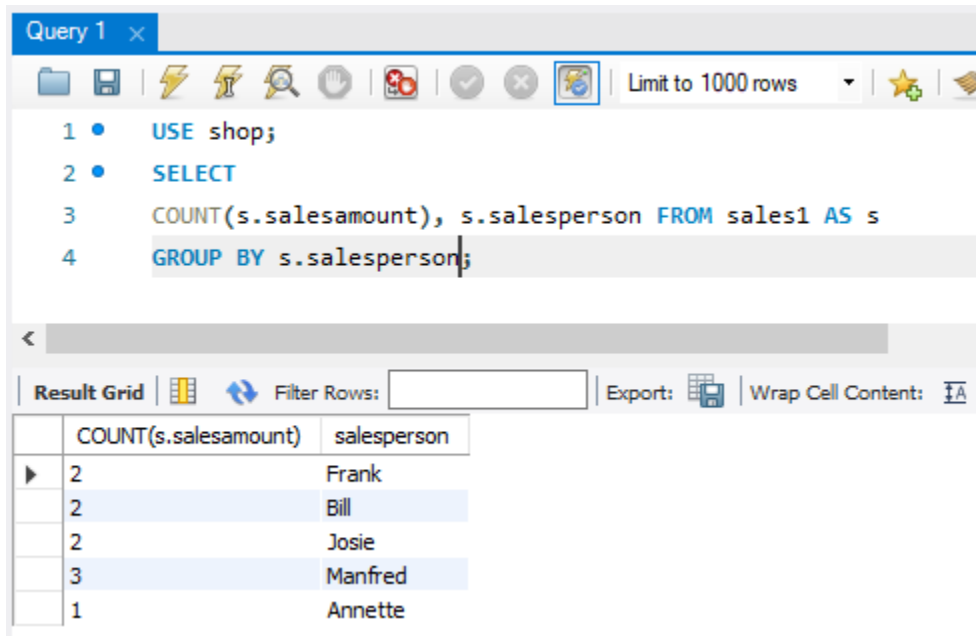
COUNT(s.salesamount)	salesperson
2	Frank
2	Bill
2	Josie
1	Annette

```

USE shop;
SELECT
COUNT(s.salesamount), s.salesperson FROM sales1 AS s
GROUP BY s.salesperson
HAVING COUNT(s.salesamount) < 3;

```

This result seems to be correct because removing the limitation "having count less than 3" results in showing 1 more result, with 3 sales:



The screenshot shows a SQL query editor window titled "Query 1". The query is as follows:

```

1 • USE shop;
2 • SELECT
3   COUNT(s.salesamount), s.salesperson FROM sales1 AS s
4   GROUP BY s.salesperson;

```

Below the query editor, the "Result Grid" is displayed, showing the results of the query. The grid has two columns: "COUNT(s.salesamount)" and "salesperson". The results are as follows:

	COUNT(s.salesamount)	salesperson
▶	2	Frank
	2	Bill
	2	Josie
	3	Manfred
	1	Annette

9. Find the number (count) of sales by each person, but only if they made less than or equal to £300 worth of sales for the past period

```

USE shop;
SELECT
COUNT(s.salesamount), s.salesperson FROM sales1 AS s
GROUP BY s.salesperson
HAVING SUM(s.salesamount) <= 300;

```

Query 1

```

1 • USE shop;
2 • SELECT
3   COUNT(s.salesamount), s.salesperson FROM sales1 AS s
4   GROUP BY s.salesperson
5   HAVING SUM(s.salesamount) <= 300;

```

Result Grid

	COUNT(s.salesamount)	salesperson
2		Frank
2		Bill
2		Josie
3		Manfred
1		Annette

This result seems correct, these people made sales of under 300 GBP:

Query 1

```

1 • USE shop;
2 • SELECT
3   COUNT(s.salesamount), SUM(s.salesamount), s.salesperson FROM sales1 AS s
4   GROUP BY s.salesperson;

```

Result Grid

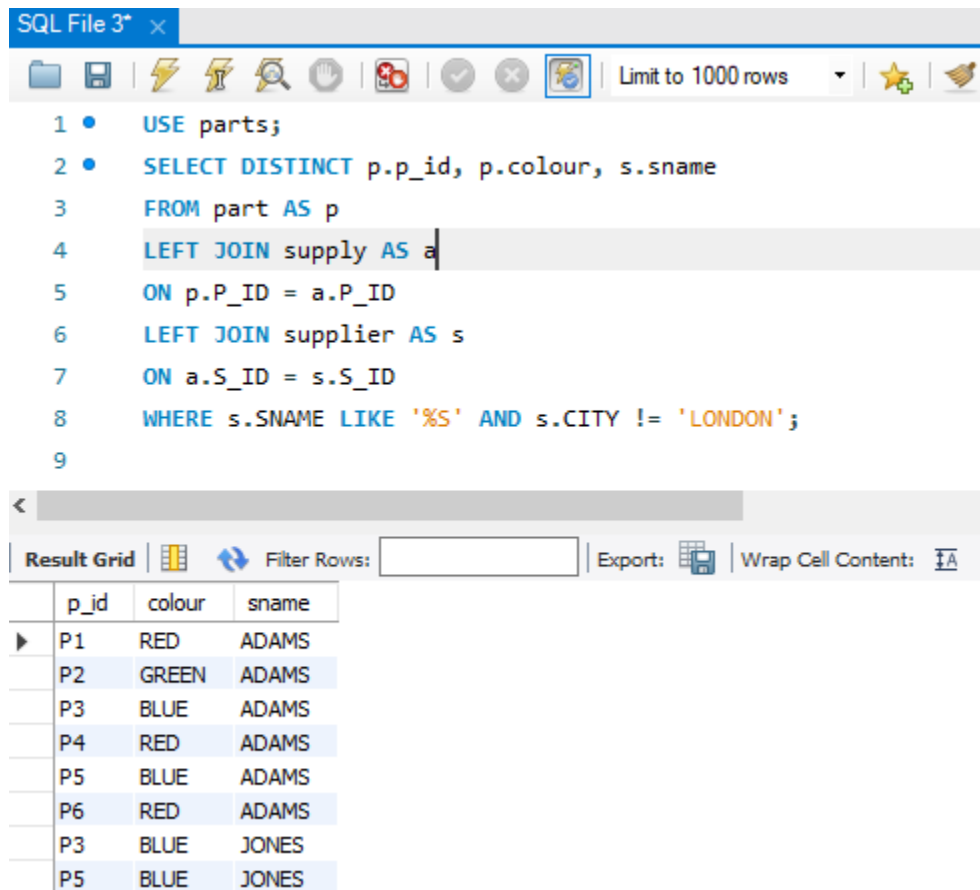
	COUNT(s.salesamount)	SUM(s.salesamount)	salesperson
2		130.57	Frank
2		172.32	Bill
2		118.29	Josie
3		206.92	Manfred
1		9.99	Annette

TASK 3

USE PARTS DB TO WRITE THE FOLLOWING QUERIES

1. Return the PartID, Colour and Supplier name, where the supplier's surname ends in an S, and the Supplier city is not London. Ensure the values are Unique.

ANSWER:



The screenshot shows a SQL editor window titled "SQL File 3*" with a toolbar and a query. The query is as follows:

```
1 • USE parts;
2 • SELECT DISTINCT p.p_id, p.colour, s.sname
3 FROM part AS p
4 LEFT JOIN supply AS a
5 ON p.P_ID = a.P_ID
6 LEFT JOIN supplier AS s
7 ON a.S_ID = s.S_ID
8 WHERE s.SNAME LIKE '%S' AND s.CITY != 'LONDON';
9
```

Below the query, the "Result Grid" is displayed, showing the results of the query. The grid has columns for p_id, colour, and sname. The results are as follows:

	p_id	colour	sname
▶	P1	RED	ADAMS
	P2	GREEN	ADAMS
	P3	BLUE	ADAMS
	P4	RED	ADAMS
	P5	BLUE	ADAMS
	P6	RED	ADAMS
	P3	BLUE	JONES
	P5	BLUE	JONES

```
USE parts;
SELECT DISTINCT p.p_id, p.colour, s.sname
FROM part AS p
LEFT JOIN supply AS a
ON p.P_ID = a.P_ID
LEFT JOIN supplier AS s
ON a.S_ID = s.S_ID
WHERE s.SNAME LIKE '%S' AND s.CITY != 'LONDON';
```

I tried using the multiquery we used in class with “IN” but it did not support 2 search values/filters, so I’ve used a multi join to first join these tables into 1 searchable table, and then applied the query limitations %S and !=LONDON as well as DISTINCT because in here due to other values in other columns (not visible here) being different, the result without “DISTINCT” gave a result with a lot of duplicates.

2. Return the supplier name, part name and project name for each case where the following conditions are true:
 - i. The supplier supplies a project with a part;
 - li. And where the supplier’s city, project city and part city are the same.

ANSWER:

```
USE parts;
SELECT s.sname, p.pname, j.jname
FROM part AS p
LEFT JOIN supply AS a
ON p.P_ID = a.P_ID
LEFT JOIN supplier AS s
ON a.S_ID = s.S_ID
LEFT JOIN project AS j
ON a.J_ID = j.J_ID
WHERE a.QUANTITY != '0' AND a.QUANTITY IS NOT NULL AND s.CITY = j.CITY AND
j.CITY = p.CITY;
```

SQL File 3*

Limit to 1000 rows

```

1 • USE parts;
2 • SELECT s.sname, p.pname, j.jname
3   FROM part AS p
4  LEFT JOIN supply AS a
5    ON p.P_ID = a.P_ID
6  LEFT JOIN supplier AS s
7    ON a.S_ID = s.S_ID
8  LEFT JOIN project AS j
9    ON a.J_ID = j.J_ID
10 WHERE a.QUANTITY IS NOT NULL AND s.CITY = j.CITY AND j.CITY = p.CITY;
11

```

Result Grid

	sname	pname	jname
▶	CLARK	COG	TAPE

Condition 1 – I used the “quantity” as “not null” as well as “not equal to zero” as means of answering the condition one = supplier supplies a project with a part – quantity of parts must then be positive, so neither missing (null) nor zero, it must be greater than zero to meet this condition.

Condition 2 – supplier city = project city and project city = part city. Checks if these values are equal – I’ve checked also with the variables uncovered, it says “LONDON” in all 3 of these cases.