

Tutorial 2 – SQL Part I

WITH SUGGESTED SOLUTIONS

Answers to Review Questions

1. Discuss the following:

1. If an SQL statement includes a GROUP BY, the attributes that can be requested in the SELECT clause will be limited. Explain that limitation.

*Ans: Limitations on attributes that can be selected when an SQL statement contains GROUP BY:
Only those columns that have a single value for each group can be included.*

2. In what order are the clauses of an SQL statement processed?

Ans: In a SELECT statement, the processing of clauses follows the order shown in Figure 6-10, and reproduced below:

- the FROM clause is evaluated first (sources of data)
- then:
 WHERE (criteria to be met, if stated), and/or
 GROUP BY (organizes rows according to values in stated column(s)), and
 HAVING (finds groups meeting criteria), and
- then SELECT (particular columns to show/display), and
- then ORDER BY (sorts rows in result table).

3. In an ORDER BY clause, what are the two ways to refer to the columns to be used for sorting the results of the query?

Ans: In an ORDER BY clause, the columns to be sorted can be referred to by: (1) column name from the actual table referenced in the FROM clause, or (2) column position in the SELECT clause list.

4. What are the different purposes of the WHERE clause and the HAVING clause? Is it possible to use both a WHERE clause and a HAVING clause in the same SELECT statement?

Ans: The WHERE clause would restrict which individual rows are included for processing by the query. The HAVING clause (used in conjunction with a GROUP BY clause) determines which summarized groups are included in the result set. Aggregate functions are not allowed in WHERE clause but they are allowed in HAVING clause.

Yes, a query may use both the WHERE and HAVING clauses.

Solutions to Problems and Exercises

The following SQL exercises are based on the Pizza database that consists of 3 tables:

MENU(pizza, price, country, base)
ITEMS(ingredient, type)

RECIPE(*pizza, ingredient*, amount)

Each row in the MENU table is uniquely identified by the pizza name (PIZZA), and contains the price (PRICE), the country of origin (COUNTRY), and the base used in making the pizza (BASE), i.e. whole meal (wm) or white flour (wf). The ITEMS table records the ingredients used for making the pizzas. Each unique ingredient (INGREDIENT) has a type (TYPE) to indicate if the ingredient is dairy, meat, fish, etc. Finally, RECIPE table records the combinations of pizzas and the ingredients used in making each pizza, also specifying the amount in grams (AMOUNT) for each main ingredient.

CREATE TABLE STATEMENTS

1. Give CREATE TABLE statements to create the corresponding PostgreSQL tables.

Further instructions:

One of the main purposes of this tutorial is for you to learn how the database (software) engine works with the tables in a database when it is executing a SELECT statement to retrieve information. This is extremely important because you will only be able to design and code a correct SELECT statement efficiently if you really understand precisely what is happening in the machine as the statement is executed.

Many students learn SQL with a Trial-and-Error (sometimes more like Hit-and-Miss) approach. This is not conducive to building a solid skill or confidence in your ability. It is also unlikely that you will develop an efficient method of constructing queries, or be able to design queries that can reliably process thousands of records in tables.

In order for you to develop this understanding, all the work will be done **manually**, with paper and pens. This class has been very specifically designed as a tutorial, not as a laboratory exercise, because in our experience, this is the best way to start developing your SQL skills.

So, please put away your laptops and mobile devices – you can, and should, practice coding SQL on a machine to your heart's content outside this tutorial and in the drop-in labs!

For the set of queries below, the tutor will go through the first question with the class on the board, writing each SELECT statement on the board, or show on the projection screen, and then they will show you how to 'trace' the execution of this statement, or pretend to be the database engine executing the query, writing down every row that will be retrieved from the table in a temporary results table, and then doing any other processing, such as sorting, of the rows/data as they write down the set of rows that form the final result (which is also a table – remember the result of a SQL statement is always a table).

Then, after the tutor has walked you through how to do this trace, working in your groups, you must 'trace' the execution of the SQL statement for the next question, which the tutor will also give to you, using the data sheets given out to you with the tutorial questions for the Pizza database.

This process is sometimes called a 'dry run' or a 'manual trace', and is commonly used in programming to check the logic and accuracy of the code.

The tutor will probably give you the SQL statement for a couple more questions so that you can trace its execution, until your group is comfortable with the process and can explain exactly what the database engine has to do for each step to retrieve the correct information from the database.

At this stage, your group can start constructing your own SELECT statements for the rest of the question. Once you have designed a SELECT statement, do a trace with it to ensure that it does give the correct results as required in the question.

Once you have done some of the first set of questions, your tutor may walk the class through tracing a more complex, aggregate or grouping query, such as those in questions 11-16, so that you can trace through your solutions for those problems, too.

As your SQL skills develop, you will not need to trace through the execution for every single row in the tables, as you will start to anticipate what the result should look like. Be careful, however, that you don't give what you're sure is the correct set of rows in the result without ensuring that your SELECT statement, as you've designed it, really does return each of those rows and that you making assumptions about its correctness...

SIMPLE QUERY STATEMENTS

- List all pizzas in alphabetic order; show all columns.

```
select *
from menu
order by pizza;
```

Result

pizza	price	country	base
americano	7.4	usa	Wm
cabanossi	7.4	italy	Wf
garlic	3.5		Wm
ham	7.3		Wf
hawaiian	7.4	hawaii	Wm
margarita	6.2	italy	Wf
mexicano	7.4	mexico	Wf
mushroom	7.3		Wm
napolitana	7.4	italy	Wf
seafood	9.2		Wm
siciliano	7.4	italy	Wm
special	9.9		Wf
stagiony	7.8	italy	Wm
vegetarian	7.4		Wm

Pizza	Price	Country	Base
* Americano	7.4	USA	Wm
cabanossi	7.4	Italy	Wf
* galic	3.5		Wm
ham	7.3		Wf
hawaiian	7.4	hawaii	Wm
margarita	6.2	Italy	Wf
* mexican	7.4	Mexico	Wf
mushroom	7.3		Wm
Napolitana	7.4	Italy	Wf
seafood	9.2		Wm
Siciliano	7.4	Italy	Wm
special	9.9		Wf
stagion	7.8	Italy	Wm
vegetarian	7.4		Wm

3. List all pizzas ordered by PRICE in descending order, and PIZZA in ascending order; show all columns.

```
select *
from menu
order by price desc, pizza;
```

Result

pizza	price	country	base
special	9.9		Wf
seafood	9.2		Wm
stagion	7.8	Italy	Wm

americano	7.4	usa	Wm
cabanossi	7.4	italy	Wf
hawaiian	7.4	hawaii	Wm
mexicano	7.4	mexico	Wf
napolitana	7.4	italy	Wf
siciliano	7.4	italy	Wm
vegetarian	7.4		Wm
ham	7.3		Wf
mushroom	7.3		Wm
margarita	6.2	italy	Wf
garlic	3.5		Wm

4. List all price categories recorded in the MENU table, eliminating duplicates.

```
select distinct price
from menu;
```

Result

```
Price
-----
3.5
6.2
7.3
7.4
7.8
9.2
9.9
```

5. List all Italian pizzas that cost less than \$7.00.

```
select pizza
from menu
where price < 7 and country = 'italy';
```

Result

```
pizza
-----
margarita
```

6. List all pizzas except those that originate from Italy or USA.

```
select pizza
from menu
where not (country = 'italy' or country = 'usa');
```

Result

```
pizza
-----
hawaiian
mexicano
```

7. Give all information for the Vegetarian, Americano, Mexicano, and Garlic pizzas.

```
select pizza, price, country, base
from menu
where pizza in ('vegetarian', 'americano', 'mexicano', 'garlic');
```

Result

pizza	price	country	base
americano	7.4	usa	Wm
mexicano	7.4	mexico	Wf
vegetarian	7.4		Wm
garlic	3.5		Wm

Note : all information refers to information of pizza in menu table only

8. List pizzas that cost between 6 and 7 dollars.

```
select pizza, price
from menu
where price between 6 and 7;
```

Result

pizza	price
margarita	6.2

9. List pizzas with name ending with the letters 'ano'.

```
select pizza
from menu
where pizza like '%ano';
```

Result

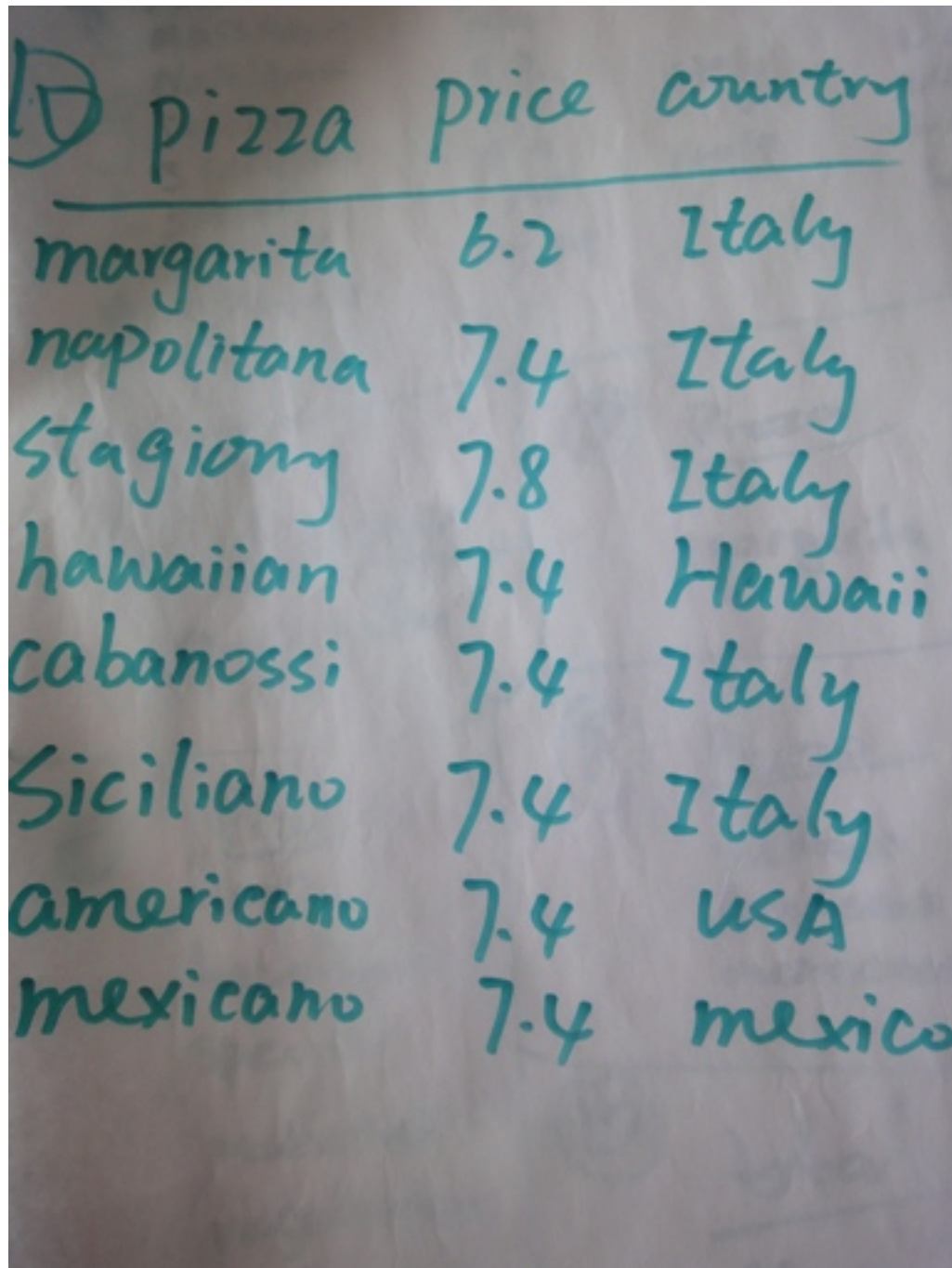
pizza
siciliano
americano
mexicano

10. List all pizzas, giving PIZZA name, PRICE and COUNTRY of origin where the COUNTRY has been specified (i.e. is not missing).

```
select pizza, price, country
from menu
where country is not null;
```

Result

pizza	price	country
margarita	6.2	italy
napolitana	7.4	italy
stagiony	7.8	italy
hawaiian	7.4	hawaii
cabanossi	7.4	italy
siciliano	7.4	italy
americano	7.4	usa
mexicano	7.4	mexico



pizza	price	country
margarita	6.2	Italy
napolitana	7.4	Italy
stagioni	7.8	Italy
hawaiian	7.4	Hawaii
cabanossi	7.4	Italy
Siciliano	7.4	Italy
americano	7.4	USA
mexicano	7.4	Mexico

11. Give the total number of rows in the menu table.

```
select count(*)
from menu;
```

Result

```
count
-----
14
```


12. How many different countries of origin are recorded in the MENU table?

```
select count(distinct country) as no_of_countries
from menu;
```

Result

```
no_of_countries
-----
4
```

13. Give the price of the cheapest Italian pizza.

```
select min(price)
from menu
where country = 'italy';
```

Result

```
min
----
6.2
```

14. Give the total price for Margarita and Vegetarian pizzas.

```
select sum(price)
from menu
where pizza in ('margarita', 'vegetarian');
```

Result

```
sum
-----
13.6
```

15. Give the average price of pizzas for each COUNTRY of origin.

```
select country, avg(price)
from menu
where country is not null
group by country;
```

Result

```
country | avg
-----+-----
usa      | 7.40000009536743
hawaii   | 7.40000009536743
mexico    | 7.40000009536743
italy     | 7.24000005722046
```

16. Give the average price of pizzas for each COUNTRY of origin; do not include countries with only

one pizza.

```
select country, avg(price)
from menu
where country is not null
group by country
having count(*) > 1;
```

Result

country	avg
Italy	7.24000005722046