# *Ejercicios SQL*

Una vez ejecutado el script pollutant.sql tendremos 3 tablas:

- Tabla Item: contiene los contaminantes que podrían ser sensados por estaciones.

```
CREATE TABLE item
(
    code integer NOT NULL,
    name VARCHAR(5) NOT NULL,
    uom VARCHAR(15) NOT NULL,
    good FLOAT ,
    normal FLOAT ,
    bad FLOAT ,
    very_bad FLOAT ,
    PRIMARY KEY (code),
    UNIQUE (name)
);
```

Su código y su nombre no puede repetirse en dicha tabla (son claves).
El atributo uom indica la unidad de medida y los otros atributos los umbrales correspondientes a límite inferiores.

Ejemplo: Hasta ahora hay 8 contaminantes de interés.

| Item (pollutant) | | | | | | |
|---|---|---|---|---|---|---|
| Code | Name | Uom | Good | Normal | Bad | Very bad |
| 1 | 'SO2' | 'ppm' | 0.02 | 0.05 | 0.15 | 1.0 |
| 3 | 'NO2' | 'ppm' | 0.03 | 0.06 | 0.2 | 2.0 |
| 5 | 'CO' | 'ppm' | 2.0 | 9.0 | 15.0 | 50.0 |
| 6 | 'O3' | 'ppm' | 0.03 | 0.09 | 0.15 | 0.5 |
| 8 | 'PM10' | 'Microgram/m3' | 30.0 | 80.0 | 150.0 | 600.0 |
| 9 | 'PM2.5' | 'Microgram/m3' | 15.0 | 35.0 | 75.0 | 500.0 |
| 11 | 'dummy' | 'ppm' | null | null | null | 1.2 |
| 12 | 'bis' | 'ppm' | null | 0.3 | 0.4 | 0.5 |

El pollutant 'SO2' se consideraría Very Bad si su valor sensado supera (>) 1.0.
Si el sensado da un valor superior (>) a 0.15 pero no supera (<=) 1.0, entonces es considerado Bad.
Si en cambio, es superior (>) a 0.05 pero no supera (<=) 0.15, se lo considera Normal.
Si fuera superior (>) 0.02 pero no supera (<=) 0.05, es Bueno.
Finalmente, si resultara que no supera (<=) 0.02 sería Excellent.

- Tabla Station: contiene las estaciones de sensado

```
CREATE TABLE station
(
    code integer NOT NULL,
    name VARCHAR(20) NOT NULL,
    address VARCHAR(100) NOT NULL,
    lat double precision NOT NULL,
    lon double precision NOT NULL,
    PRIMARY KEY (code),
    UNIQUE (lat, lon)
);
```

Su código no puede repetirse en dicha tabla. Tampoco la combinación lat+lon (son claves).

Ejemplo: Hasta ahora hay 3 estaciones.

| Station | | | | |
|---|---|---|---|---|
| Code | Name | address | Lat | lon |
| 101 | 'Jongno-gu' | '19 Jong-ro 35ga-gil, Jongno-gu, Seoul, Republic of Korea' | 37.57201639 | 127.0050074 |
| 102 | 'Jung-gu' | '15; Deoksugung-gil, Jung-gu, Seoul, Republic of Korea' | 37.56426289 | 126.9746756 |
| 103 | 'Yongsan-gu' | '136; Hannam-daero, Yongsan-gu, Seoul, Republic of Korea' | 37.54003270 | 127.00485 |

- Tabla Measurement: contiene las mediciones en diferentes momentos. Es decir, es un histórico del valor de cierta polución medida por cierta estación en cierto momento. No se indica la unidad de medida porque eso está indicado en la tabla Item. Asocia ítem_code con station_code con cierto date.

```
CREATE TABLE measurement
(
    date timestamp NOT NULL,
    station_code integer NOT NULL,
    item_code integer NOT NULL,
    value float NOT NULL,
    PRIMARY KEY (date, station_code, item_code),
    FOREIGN KEY (item_code)
        REFERENCES item ON DELETE CASCADE,
    FOREIGN KEY (station_code)
        REFERENCES station ON DELETE CASCADE
);
```

Ejemplo: Hasta ahora hay 53 mediciones históricas.

Notar que el ítem_code 1 ('SO2') fue medido por la estación 101 ('Jongno-gu') varias veces, pero en diferentes instantes de tiempo.

| Measurement | | | |
|---|---|---|---|
| Date | Station_code | Item_code | value |
| '2017-01-01 00:00' | 101 | 1 | 0.04 |
| '2017-01-01 00:00' | 101 | 3 | 0.590 |
| '2017-01-01 00:00' | 101 | 5 | 2.5 |
| '2017-01-01 00:00' | 101 | 6 | 0.2 |
| '2017-01-01 00:00' | 101 | 8 | 93.0 |
| '2017-01-01 00:00' | 101 | 9 | 7.0 |
| '2017-01-01 00:00' | 102 | 1 | 0.6 |
| '2017-01-01 00:00' | 102 | 3 | 0.068 |
| '2017-01-01 00:00' | 102 | 5 | 1.3 |
| '2017-01-01 00:00' | 102 | 6 | 0.002 |
| '2017-01-01 00:00' | 102 | 8 | 77.0 |
| '2017-01-01 00:00' | 102 | 9 | 63.0 |
| '2017-01-01 00:00' | 103 | 1 | 0.005 |
| '2017-01-01 00:00' | 103 | 3 | 0.039 |
| '2017-01-01 00:00' | 103 | 5 | 1.4 |
| '2017-01-01 00:00' | 103 | 6 | 0.002 |
| '2017-01-01 00:00' | 103 | 8 | 70.0 |
| '2017-01-01 00:00' | 103 | 9 | 68.0 |
| '2017-01-01 01:00' | 101 | 1 | 0.004 |
| '2017-01-01 01:00' | 101 | 3 | 0.579 |
| '2017-01-01 01:00' | 101 | 5 | 1.2 |
| '2017-01-01 01:00' | 101 | 6 | 0.002 |
| '2017-01-01 01:00' | 101 | 8 | 71.0 |
| '2017-01-01 01:00' | 101 | 9 | 59.0 |
| '2017-01-01 01:00' | 102 | 1 | 0.006 |

| | | | |
|---|---|---|---|
| '2017-01-01 01:00' | 102 | 3 | 0.066 |
| '2017-01-01 01:00' | 102 | 5 | 1.4 |
| '2017-01-01 01:00' | 102 | 6 | 0.002 |
| '2017-01-01 01:00' | 102 | 8 | 76.0 |
| '2017-01-01 01:00' | 102 | 9 | 63.0 |
| '2017-01-01 01:00' | 103 | 1 | 0.004 |
| '2017-01-01 01:00' | 103 | 3 | 0.038 |
| '2017-01-01 01:00' | 103 | 5 | 1.4 |
| '2017-01-01 01:00' | 103 | 6 | 0.002 |
| '2017-01-01 01:00' | 103 | 8 | 73.0 |
| '2017-01-01 02:00' | 101 | 1 | 0.004 |
| '2017-01-01 02:00' | 101 | 3 | 0.0559 |
| '2017-01-01 02:00' | 101 | 5 | 1.2 |
| '2017-01-01 02:00' | 101 | 6 | 0.002 |
| '2017-01-01 02:00' | 101 | 8 | 70.0 |
| '2017-01-01 02:00' | 101 | 9 | 59.0 |
| '2017-01-01 02:00' | 102 | 1 | 0.005 |
| '2017-01-01 02:00' | 102 | 3 | 0.063 |
| '2017-01-01 02:00' | 102 | 5 | 1.2 |
| '2017-01-01 02:00' | 102 | 6 | 0.002 |
| '2017-01-01 02:00' | 102 | 8 | 73.0 |
| '2017-01-01 02:00' | 102 | 9 | 57.0 |
| '2017-01-01 02:00' | 103 | 1 | 0.005 |
| '2017-01-01 02:00' | 103 | 3 | 0.037 |
| '2017-01-01 02:00' | 103 | 5 | 1.4 |
| '2017-01-01 02:00' | 103 | 6 | 0.002 |
| '2017-01-01 02:00' | 103 | 8 | 67.0 |
| '2017-01-01 02:00' | 103 | 9 | 65.0 |

Importante: si un item no fue medido por cierta estación NO APARECE en la tabla measurement (ni siquiera con valor 0)

## *Ejercicio 1 (clausula SELECT FROM ORDER BY)*

1.1) La tabla measurement tiene 53 mediciones. Tiene el ítem_code pero no el nombre del contaminante. Para poder listar el nombre de dicho contaminante se precisa usar la información de 2 tablas: measurement y item.

¿Por qué la siguiente consulta no arroja lo esperado, es decir, no muestra 53 valores una columna más con el nombre del contaminante? ¿Por qué se obtienen 424 ( 53 * 8) tuplas?
Explicar

```
SELECT  measurement.*, name
FROM measurement,  item
```

Rta

Porque el producto cartesiano combina todas las tuplas de la tabla measurement con todas las tuplas de la tabla item sin ninguna restricción.

Para poder entender el resultado, vamos a mostrar también el code de ítem, es decir:

```
SELECT  measurement.*, name, code
FROM measurement,  item
```

El resultado obtenido (lo mostramos parcialmente) es:

| date | station_code | item_code | value | name | code |
|---|---|---|---|---|---|
| 2017-01-01T00:00:00Z | 101 | 1 | 0.04 | SO2 | 1 |
| 2017-01-01T00:00:00Z | 101 | 3 | 0.5900000000000001 | SO2 | 1 |
| 2017-01-01T00:00:00Z | 101 | 5 | 2.5 | SO2 | 1 |
| 2017-01-01T00:00:00Z | 101 | 6 | 0.2 | SO2 | 1 |
| 2017-01-01T00:00:00Z | 101 | 8 | 93 | SO2 | 1 |

Las primeras 4 columnas fueron tomadas de la tabla measurement , las últimas de la tabla ítem.
Pero en cada tupla, el "ítem_code" no coincide con el "code" del pollutant esperado.

No puede ser que ítem_code 1 diga que el pollutant es SO2, y el 3 también y que más adelante el 1 diga que es NO2, etc

Debemos agregar condición de junta. Si ejecutáramos

```
SELECT  measurement.*, name,  code
FROM measurement JOIN  item  ON  item_code = code
```

Obtendríamos las 53 tuplas esperadas. Mostramos solo un subconjunto del resultset:

| date | station_code | item_code | value | name | code |
|---|---|---|---|---|---|
| 2017-01-01T00:00:00Z | 101 | 1 | 0.04 | SO2 | 1 |
| 2017-01-01T00:00:00Z | 101 | 3 | 0.5900000000000001 | NO2 | 3 |
| 2017-01-01T00:00:00Z | 101 | 5 | 2.5 | CO | 5 |
| 2017-01-01T00:00:00Z | 101 | 6 | 0.2 | O3 | 6 |
| 2017-01-01T00:00:00Z | 101 | 8 | 93 | PM10 | 8 |
| 2017-01-01T00:00:00Z | 101 | 9 | 7 | PM2.5 | 9 |
| 2017-01-01T00:00:00Z | 102 | 1 | 0.6 | SO2 | 1 |

Los valores SIEMPRE coinciden en el código, y por lo tanto el "name" es ahora correcto para cada medición.

1.2) Escribir la consulta en forma correcta. En el ejemplo, deberían obtenerse 53 tuplas. Las mismas de la tabla measurement pero con una columna adicional que indique el nombre del contaminante.

Rta

Ya no hace falta mostrar code de la tabla ítem. Solo habíamos usado ese valor para asegurarnos que la junta entre tablas es la correcta y restringir sus valores:

```
SELECT  measurement.*, name
FROM measurement JOIN  item  ON  item_code = code
```

1.3) Mostrar del histórico de mediciones solo hasta las 8 más recientes, desplegando el nombre del ítem también

Rta

```
SELECT  measurement.*, name, code
FROM measurement JOIN  item  ON  item_code = code
ORDER BY date DESC
LIMIT 8
```

1.4) Escribir una consulta SQL que permita obtener: nombre de la estación (no código), nombre del contaminante (no código), fecha de la medición y el valor medido, mostrando primero las más reciente y de repetirse la fecha, ordenar luego por nombre del ítem ascendentemente y luego por nombre de la estación ascendentemente. Con los datos del ejemplo, deberíamos obtener sólo 53 tuplas. Mostramos solo un subconjunto

| name | name | date | value |
|---|---|---|---|
| Jongno-gu | CO | 2017-01-01T02:00:00Z | 1.2 |
| Jung-gu | CO | 2017-01-01T02:00:00Z | 1.2 |
| Yongsan-gu | CO | 2017-01-01T02:00:00Z | 1.4 |
| Jongno-gu | NO2 | 2017-01-01T02:00:00Z | 0.055999999999999994 |
| Jung-gu | NO2 | 2017-01-01T02:00:00Z | 0.063 |
| Yongsan-gu | NO2 | 2017-01-01T02:00:00Z | 0.037000000000000005 |

Rta Notar que no podemos ejecutar

```
SELECT name, name, date, value
FROM (measurement JOIN item ON item_code = code) JOIN station
      ON station_code= code
ORDER BY date DESC, name ASC, name ASC
```

porque ahora que combinamos las 3 tablas, el atributo "code" está tanto en la tabla STATION como en ITEM. Por lo tanto, resulta ambiguo a cuál nos estamos refiriendo.
Ídem con el atributo "name".

Para desambiguar, debemos prefijar la columna que resulta ambigua con el nombre de la tabla a la cual nos referimos.

```
SELECT station.name, item.name, date, value
FROM (measurement JOIN item ON item_code = item.code) JOIN station
      ON station_code= station.code
ORDER BY date DESC, item.name ASC, station.name ASC
```

1.5) Ídem al anterior, pero se espera que en el resultado los nombres de las columnas se muestren diferentes. El nombre de la estación debe aparecer como "EstacionNombre", el del ítem "Contaminante", el de la fecha "Instante". El valor no cambiarlo.

Así cambiaría el header:

| estacionnombre | contaminante | instante | value |
|---|---|---|---|

Rta En general, puede omitirse la palabra AS.

```
SELECT station.name AS EstacionNombre, item.name AS Contaminante, date AS Instante, value
FROM (measurement JOIN item ON item_code = item.code) JOIN station
                        ON station_code= station.code
ORDER BY date DESC, item.name ASC, station.name ASC
```

## Ejercicio 2 (Uso de WHERE)

2.1) La misma consulta que realizamos en el item 1.4

```
SELECT name, name, date, value
FROM (measurement JOIN item ON item_code = code) JOIN station
      ON station_code= code
ORDER BY date DESC, name ASC, name ASC
```

podría realizarse colocando la condición de "matcheo" en la cláusula WHERE. Es decir, usar un producto cartesiano entre dichas 3 tablas, para luego establecer en el WHERE las restricciones esperadas.
Reescribir dicha consulta de esta manera

Rta

```
SELECT station.name, item.name, date
FROM measurement, item, station
WHERE item_code = item.code AND station_code= station.code
ORDER BY date DESC, item.name ASC, station.name ASC
```

2.2) Mostrar los nombres de los **contaminantes cuyos valores registrados** fueron Very Bad (superaron el valor VeryBad de la tabla indicada por ITEM). Mostrar dicho valor y el valor del umbral correspondiente (el cual depende de cada item)

Con los datos del ejemplo no se obtiene ninguno

Rta

```
SELECT item.name, value, very_bad
from measurement, item
WHERE item_code = item.code AND value > very_bad
```

2.3) Mostrar los nombres de los contaminantes cuyos valores registrados fueron Bad (superaron el valor Bad pero no el valor VeryBad de los umbrales indicados por la tabla ITEM). Mostrar dicho valor y los 2 umbrales correspondientes.

Con los datos del ejemplo se obtendría

| name | value | bad | very_bad |
|---|---|---|---|
| NO2 | 0.5900000000000001 | 0.2 | 2 |
| O3 | 0.2 | 0.15 | 0.5 |
| SO2 | 0.6 | 0.15 | 1 |
| NO2 | 0.58 | 0.2 | 2 |

Rta

```
SELECT item.name, value, bad as limiteinferior, very_bad  as limitesuperior
FROM measurement, item
WHERE item_code = item.code AND value > bad AND value <= very_bad
```

2.4) Mostrar el nombre de los contaminantes que no se miden en unidades ppm

Con los datos del ejemplo sólo debería obtenerse 2 tuplas.

| name |
|---|
| PM10 |
| PM2.5 |

Rta

```
SELECT name
from item
WHERE uom <> 'ppm'
```

Si lo consultamos en mayúsculas no lo encontraría porque fue ingresado en minúscula. Cuidado con eso. Sino preguntar así:

```
SELECT name
FROM item
WHERE Upper(uom) <> 'PPM'
```

O así

```
SELECT name
FROM  item
WHERE lower(uom) <> 'ppm'
```

2.5) Mostrar aquellos nombres de contaminantes que todavía no tienen ingresados algún valor de sus umbrales.
Obtendremos 2: dummy y bis

Rta

Notar que esta consulta es incorrecta y de hecho devuelve 0 tuplas

```
SELECT name
FROM item
WHERE good = null OR normal = null OR bad = null OR very_bad= null
```

porque al preguntar si un valor de cierto atributo es = null or <> null no devuelve true. Recordar que null se usa para representar diferentes cosas.

La forma correcta de preguntar por null es con el predicado IS NULL (o IS NOT NULL). Es decir,

```
SELECT name
FROM item
WHERE good IS null OR normal IS null OR bad IS null OR very_bad IS NULL
```

así obtendríamos las 2 tuplas que poseen algún valor no definido/desconocido o que no aplica

2.6) Mostrar aquellos código y nombres de contaminantes que todavía no tienen mediciones asociadas
Con los datos del ejemplo debería obtenerse

Query #1 **Execution time: 1ms**

| code | name |
|---|---|
| 11 | dummy |
| 12 | bis |

Rta

```
SELECT code, name
FROM item
WHERE code NOT IN (SELECT item_code FROM measurement)
```

2.7) Notar que para ver cuáles son los contaminantes (código y nombre) que sí tuvieron mediciones podemos proceder de diferentes formas también:

a) con algún tipo de Join
b) con un IN

Realizar las 2 versiones. Debe obtenerse:

| code | name |
|---|---|
| 1 | SO2 |
| 3 | NO2 |
| 5 | CO |
| 6 | O3 |
| 8 | PM10 |
| 9 | PM2.5 |

Rta
a) es necesario pedir DISTINCT para quitar repetidos si usamos Join

```
SELECT DISTINCT  code, name
 FROM item, measurement
 WHERE code= item_code
ORDER BY code
```

b) versión IN no precisa DISTINCT porque en la tabla ITEM los contaminantes no se repiten

```
SELECT  code, name
FROM item
WHERE code IN (SELECT item_code FROM measurement)
ORDER BY code
```

2.8) Mostrar los nombres de contaminantes que tuvieron algún valor registrado considerado excelente (<= good), ordenados por su código numérico

Con los datos del ejemplo obtendría

| name |
|---|
| SO2 |
| CO |
| O3 |
| PM2.5 |

Rta

```
SELECT  name
FROM item
WHERE code IN (SELECT item_code FROM measurement WHERE value <= good)
ORDER BY code
```

2.9) Reescribir la consulta anterior para no solo mostrar el nombre del contaminante sino también cuál es ese valor que lo hizo excelente en dicha medición. Para verificar el resultado, mostrar también el valor del umbral Good

Deberíamos obtener

| name | medido | umbralexcelente |
|---|---|---|
| SO2 | 0.005 | 0.02 |
| SO2 | 0.006 | 0.02 |
| SO2 | 0.005 | 0.02 |
| SO2 | 0.005 | 0.02 |
| SO2 | 0.004 | 0.02 |
| SO2 | 0.004 | 0.02 |
| SO2 | 0.004 | 0.02 |

| | | |
|---|---|---|
| CO | 1.4 | 2 |
| CO | 1.3 | 2 |
| CO | 1.2 | 2 |
| CO | 1.2 | 2 |
| CO | 1.2 | 2 |
| CO | 1.4 | 2 |
| CO | 1.4 | 2 |
| CO | 1.4 | 2 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| O3 | 0.002 | 0.03 |
| PM2.5 | 7 | 15 |

Rta. Notar que si no coloco también la tabla MEASUREMENT en el FROM solo podría mostrar el umbral, pero no el valor registrado. Es decir, como mucho puedo hacer

```
SELECT  name, good
FROM item
WHERE code IN (SELECT item_code FROM measurement WHERE value <= good)
ORDER BY code
```

Para resolver lo pedido preciso: involucrar en el FROM la tabla ITEM porque preciso el nombre y también la tabla MEASUREMENT, porque preciso el valor medido. Es decir:

```
SELECT  name, value AS medido, good AS umbralExcelente
FROM item, measurement
WHERE code= item_code and value <= good
ORDER BY code
```

2.10) Mostrar los nombres de las estaciones que siempre midieron contaminantes que dieron valores aceptables, es decir TODAS sus mediciones arrojaron excelente, good o normal.

Con los valores del ejemplo debería obtenerse

| name |
| --- |
| Yongsan-gu |

Rta

Si quisiéramos calcular cuales son los códigos de las estaciones que alguna vez midieron contaminantes bad o very_bad, la consulta sería

```
SELECT station_code, name, value, bad
FROM item , measurement
WHERE code = item_code AND value >bad
```

y obtendríamos la estación 101 y 102

| station_code | name | value | bad |
| --- | --- | --- | --- |
| 101 | NO2 | 0.5900000000000001 | 0.2 |
| 101 | O3 | 0.2 | 0.15 |
| 102 | SO2 | 0.6 | 0.15 |
| 101 | NO2 | 0.58 | 0.2 |

Para obtener las estaciones que no figuran en esta tabla, deberíamos hacer un SELECT ANIDADO con NOT IN

```
SELECT name
FROM station
WHERE code NOT IN
  (SELECT station_code
  FROM item , measurement
  WHERE code = item_code AND value >bad)
```

Notar que la siguiente consulta es errónea porque muestra estaciones que “alguna vez dieron aceptable”.

INCORRECTA:

```
SELECT name
FROM station
WHERE code IN
  (SELECT station_code
  FROM item , measurement
  WHERE code = item_code AND value <= bad)
```

Lo que se pedía era que TODAS sus mediciones fueran aceptables.

**Ejercicio 3 (Funciones de Agregación y Clausula Group by/ Having)**

3.1) Mostrar el máximo valor registrado en las mediciones

Verificarlo realizando un SELECT ORDER BY DESC con LIMIT 1

Rta

```
SELECT max(value)
FROM measurement
```

Coincide con

```
SELECT value
FROM measurement
ORDER BY value DESC
LIMIT 1
```

3.2) Mostrar cuántos son los contaminantes que tienen alguna medición. Con los datos del ejemplo deberíamos obtener 6

Rta
Esta consulta es incorrecta porque los contaminantes aparecen repetidos en el histórico

```
SELECT count(item_code)
FROM measurement
```

Para que sea correcta deberíamos realizar

```
SELECT count(DISTINCT  item_code)
FROM measurement
```

3.3) Calcular por cada estación, cuántas mediciones ha realizado. Es decir, la distribución de mediciones entre las estaciones. Se espera que se despliegue el nombre de la estación junto a dicha cantidad.

Con los datos del ejemplo deberíamos obtener

| name | cantidad |
|---|---|
| Yongsan-gu | 17 |
| Jung-gu | 18 |
| Jongno-gu | 18 |

Rta

```
SELECT  name, count(*) as cantidad
FROM station, measurement
WHERE code= station_code
GROUP BY code, name
```

3.4) Ha medido alguna estación más de un contaminante? Mostrar por cada estación cuántos contaminantes diferentes a medido.

Con los datos del ejemplo, debería obtenerse

| name | cantidad |
|---|---|
| Jongno-gu | 6 |
| Jung-gu | 6 |
| Yongsan-gu | 6 |

Rta
La opción correcta es

```
SELECT  name, count(DISTINCT item_code) AS cantidad
FROM station, measurement
WHERE code= station_code
GROUP BY code, name
```

Notar que esta consulta es incorrecta porque cuenta al mismo contaminante varias veces por participar de varias mediciones en el tiempo

```
SELECT  name, count(item_code) as cantidad
FROM station, measurement
WHERE code= station_code
GROUP BY code, name
```

3.5) Mostrar los nombres de los contaminantes cuyo valor promedio de sus mediciones fue mayor que 10. Mostrar nombre y dicho promedio.

Con los datos del ejemplo debería obtenerse

| name | promedio |
|---|---|
| PM10 | 74.44444444444444 |
| PM2.5 | 55.125 |

Rta

```
SELECT  name, avg(value) AS promedio
FROM item, measurement
WHERE code= item_code
GROUP BY  code, name
HAVING AVG(value) > 10
```

3.6) Mostrar aquellas estaciones que en un mismo instante de tiempo midieron más de un contaminante. Desplegar el nombre de dicha estación, la fecha y dicha cantidad.

Con los datos del ejemplo deberíamos obtener

| name | date | qty |
|---|---|---|
| Jung-gu | 2017-01-01T02:00:00Z | 6 |
| Yongsan-gu | 2017-01-01T02:00:00Z | 6 |
| Jongno-gu | 2017-01-01T02:00:00Z | 6 |
| Jongno-gu | 2017-01-01T00:00:00Z | 6 |
| Yongsan-gu | 2017-01-01T01:00:00Z | 5 |
| Jung-gu | 2017-01-01T01:00:00Z | 6 |
| Jung-gu | 2017-01-01T00:00:00Z | 6 |
| Yongsan-gu | 2017-01-01T00:00:00Z | 6 |
| Jongno-gu | 2017-01-01T01:00:00Z | 6 |

Rta

```
SELECT  station.name, DATE, count( item_code) AS qty
FROM station , measurement
WHERE station.code= station_code
GROUP BY station.code, station.name, date
HAVING COUNT( item_code) > 1
```

3.7) Se quiere obtener la misma agregación, pero por año. Es decir

| name | date_part | qty |
|---|---|---|
| Jongno-gu | 2017 | 18 |
| Yongsan-gu | 2017 | 17 |
| Jung-gu | 2017 | 18 |

La forma de obtener solo el año de una fecha es con la funcion extract:
extract(year from valorfecha)

Explicar por qué al ejecutar la siguiente consulta ni siquiera compila

```
SELECT  station.name, DATE, count( item_code) AS qty
FROM station , measurement
WHERE station.code= station_code
GROUP BY station.code, station.name, extract(year from date)
HAVING COUNT( item_code) > 1
```

¿Cómo debería escribirse para averiguar cuantos ítems diferentes midió una misma estación durante un mismo año?

Rta LO que se coloca en el group by es lo que puede colocarse en el SELECT sin agregar.

```
SELECT  station.name, extract(year from date) as year, count( item_code) AS qty
FROM station , measurement
WHERE station.code= station_code
GROUP BY station.code, station.name, extract(year from date)
HAVING COUNT( item_code) > 1
```

3.8) Se quiere obtener para cada nombre de contaminante la máxima y mínima cantidad registrada. Pero si un contaminante todavía no registra mediciones, igual se lo quiere obtener en el listado, con valor 0

Con el ejemplo, la consulta devolvería

| name | coalesce | min |
|---|---|---|
| dummy | 0 | 0 |
| bis | 0 | 0 |
| CO | 2.5 | 1.2 |
| PM10 | 93 | 67 |
| O3 | 0.2 | 0.002 |
| SO2 | 0.6 | 0.004 |
| NO2 | 0.5900000000000001 | 0.037000000000000005 |
| PM2.5 | 68 | 7 |

Rta
La siguiente consulta logra conservar las tuplas que no matchean, pero no el resultado esperado. El left outer join nos permite conservar tuplas que no matchean, pero para MAX y MIN se obtiene NULL y no O

```
SELECT  item.name, max(value), min(value)
FROM item left outer join measurement on item.code= item_code
GROUP BY item.code, item.name
```

La consulta debería transformar el NULL en otro valor. Para eso puede usarse la función COALESCE( expresión, valor1, ….) que evalúa los operandos de izquierda a derecha (primero expresión, luego valor1, etc) hasta encontrar un valor que no de NULL.

```
SELECT  item.name, COALESCE(max(value), 0) , min( COALESCE(value, 0) )
FROM item left outer join measurement on item.code= item_code
GROUP BY item.code, item.name
```

En este caso da igual si primero se aplica COALESCE y luego la función de agregación o viceversa.