Notes from doing hacker rank:
useful SQL syntax:
CHAR_LENGTH(): string length
MAX() / MIN(): max or min or a number or a string
LEFT(string, number): select the first several chars from a string
RIGHT(string, number): select the right several chars from a string
MOD(int_a, int_b): calculate int_a%int_b
AS: select a as b: rename column after selecting

Notes from SQLZOO: (simply some codes written by me)

SELECT name, continent, population FROM world

————————————

基本语句
SELECT name FROM world
WHERE population >= 200000000

————————————

select时可以对变量做加减乘除运算，无需事先生成新变量
SELECT name, gdp/population FROM world
WHERE population >= 200000000

————————————

SELECT name, population/1000000 FROM world
WHERE continent = 'South America'

————————————

SELECT name, population FROM world
WHERE name IN ('France','Germany','Italy')

SELECT * FROM table
WHERE id IN (3,4,8)

————————————

模糊匹配
SELECT name FROM world
WHERE name LIKE '%United%'

————————————

SELECT name, population, area FROM world
WHERE area>3000000 OR population>250000000

————————————

exclusive OR (两个条件有且只有一个为真)
SELECT name, population, area FROM world
WHERE area>3000000 XOR population>250000000

————————————

用ROUND function取小数点后两位

```
SELECT name, ROUND(population/1000000, 2), ROUND(gdp/1000000000, 2) FROM world
WHERE continent = 'South America’
```

————————————

ROUND the output value to the nearest 1000

```
SELECT name, ROUND(gdp/population, -3) FROM world
WHERE gdp>=1000000000000
```

————————————

use LENGTH function to calculate String length

```
SELECT name, capital FROM world
 WHERE LENGTH(name)=LENGTH(capital)
```

————————————

左起第1个字符：LEFT(name, 1)　不等于：　<>

```
SELECT name, capital FROM world
WHERE  LEFT(name,1)=LEFT(capital,1) AND name<>capital
```

————————————

选择不含空格的string：用NOT LIKE

```
SELECT name
   FROM world
WHERE name NOT LIKE '% %’
```

————————————

```
SELECT yr, subject, winner FROM nobel
WHERE yr=1980 AND subject NOT IN ('Chemistry', ‘Medicine')
```

————————————

降序排列所选项目

```
SELECT winner, yr, subject FROM nobel
WHERE winner LIKE ('Sir %')
ORDER BY yr DESC
```

————————————

Nested SELECT

List each country name where the population is larger than that of 'Russia'.

```
SELECT name FROM world
  WHERE population >
     (SELECT population FROM world
      WHERE name=‘Russia')
```

————————————

Show the countries in Europe with a per capita GDP greater than 'United Kingdom'.

```
SELECT name FROM world
WHERE gdp/population >
(SELECT gdp/population FROM world
WHERE name = 'United Kingdom')
AND continent = ‘Europe'
```

———————

List the name and continent of countries in the continents containing either Argentina or Australia. Order by name of the country.

```
SELECT name, continent FROM world WHERE continent IN
(SELECT continent FROM world WHERE name IN ('Argentina', 'Australia'))
ORDER BY name
```

———————

Which country has a population that is more than Canada but less than Poland? Show the name and the population.

```
SELECT name, population FROM world
WHERE population > (SELECT population FROM world WHERE name = 'Canada')
AND population < (SELECT population FROM world WHERE name = 'Poland')
```

———————

Show the name and the population of each country in Europe. Show the population as a percentage of the population of Germany.

```
SELECT name, CONCAT(ROUND(population*100/(SELECT population FROM world WHERE
name = 'Germany'),0),'%')
FROM world WHERE continent = ‘Europe'
```

———————

Which countries have a GDP greater than every country in Europe? [Give the name only.] (Some countries may have NULL gdp values)

```
SELECT name FROM world WHERE gdp>
(SELECT MAX(gdp) FROM world WHERE continent = ‘Europe')
```

———————

correlated subqueries

Find the largest country (by area) in each continent, show the continent, the name and the area:

```
SELECT continent, name, area FROM world x
  WHERE area >= ALL
    (SELECT area FROM world y
       WHERE y.continent=x.continent
         AND area>0)
```

A correlated subquery works like a nested loop: the subquery only has access to rows related to a single record at a time in the outer query. The technique relies on table aliases to identify two different uses of the same table, one in the outer query and the other in the subquery.

One way to interpret the line in the **WHERE** clause that references the two table is *"… where the correlated values are the same"*.

In the example provided, you would say *"select the country details from world where the population is greater than or equal to the population of all countries where the continent is the same"*.

———————

List each continent and the name of the country that comes first alphabetically.

```
SELECT continent, name FROM world x
WHERE name<= ALL
(SELECT name FROM world y WHERE y.continent = x.continent)
```

———————

Find the continents where all countries have a population <= 25000000. Then find the names of the countries associated with these continents. Show name, continent and population.

```
SELECT name, continent, population FROM world x WHERE 25000000>=
ALL (SELECT population FROM world y WHERE y.continent = x.continent)
```

———————

Some countries have populations more than three times that of any of their neighbours (in the same continent). Give the countries and continents.

```
SELECT name, continent FROM world x
WHERE (population/3) > ALL (SELECT population FROM world y WHERE y.continent =
x.continent AND y.name!=x.name)
```

———————

```
SELECT SUM(population)
FROM world
```

———————

```
SELECT DISTINCT Step
FROM salary_range_by_job_classification
WHERE Job_Code BETWEEN '0110' AND '0400'
```

———————

```
SELECT DISTINCT continent FROM world
```

———————————

```
SELECT SUM(gdp) FROM world WHERE continent = ‘Africa'
```

———————————

```
SELECT COUNT(name) FROM world WHERE area >=1000000
```

———————————

```
SELECT SUM(population) FROM world WHERE name IN ('Estonia', 'Latvia', ‘Lithuania')
```

———————————

```
SELECT continent, COUNT(name) FROM world GROUP BY continent
```

———————————

```
SELECT continent, COUNT(name) FROM world
WHERE population>10000000
GROUP BY continent
```

———————————

evaluate condition after group by

```
SELECT continent FROM world
GROUP BY continent
HAVING SUM(population)>=100000000
```

Note: WHERE filters before data is grouped; HAVING filters after data is grouped
Rows eliminated by the WHERE clause will not be included in the group

———————————

CASE WHEN

```
  CASE WHEN condition1 THEN value1
       WHEN condition2 THEN value2
       ELSE def_value
  END
```

CASE allows you to return different values under different conditions.
If there no conditions match (and there is not ELSE) then NULL is returned.

```
SELECT name, population
     ,CASE WHEN population<1000000
          THEN 'small'
          WHEN population<10000000
          THEN 'medium'
          ELSE 'large'
       END
  FROM bbc
```

———————————

JOIN

```
SELECT *
  FROM game JOIN goal ON (id=matchid)
```

———————————

```
SELECT player,teamid, stadium, mdate
  FROM game JOIN goal ON (id=matchid) WHERE teamid='GER'
```

———————————

```
SELECT team1, team2, player FROM game JOIN goal ON (id = matchid)
WHERE player LIKE 'Mario%'
```

———————————

```
SELECT player, teamid, coach, gtime
  FROM goal JOIN eteam ON (teamid=id)
 WHERE gtime<=10
```

———————————

```
SELECT mdate, teamname FROM game JOIN eteam ON (team1 = eteam.id)
WHERE coach = 'Fernando Santos'
```

———————————

```
SELECT player FROM game JOIN goal ON (id = matchid)
WHERE stadium = 'National Stadium, Warsaw'
```

———————————

```
SELECT DISTINCT player FROM game JOIN goal ON (id = matchid)
    WHERE (team1='GER' OR team2='GER') AND teamid!='GER'
```

———————————

```
SELECT teamname, COUNT(teamid)
  FROM eteam JOIN goal ON (id=teamid)
 GROUP BY teamname
```

———————————

```
SELECT stadium, COUNT(gtime) FROM game JOIN goal ON (id = matchid)
GROUP BY stadium
```

———————————

```
SELECT matchid, mdate, COUNT(gtime)
  FROM game JOIN goal ON (id = matchid)
 WHERE (team1 = 'POL' OR team2 = 'POL')
GROUP BY matchid, mdate
```

———————————

```
SELECT matchid, mdate, COUNT(gtime)
FROM game JOIN goal ON (id = matchid)
WHERE teamid = 'GER'
GROUP BY matchid, mdate
```

———————————

Joining multiple tables, and 对table name使用缩写

```
SELECT o.OrderID, c.CompanyName, e.LastName
FROM ( (Orders o INNER JOIN Customers c ON
o.CustomerID = c.CustomerID)
INNER JOIN Employees e ON o.EmployeeID =
e.EmployeeID)
```

Joining within a range using between

```
SELECT (CASE WHEN g.grade < 8 THEN 'NULL' ELSE s.name END),
g.grade, s.marks
FROM students s INNER JOIN grades g ON s.marks BETWEEN g.min_mark AND g.max_mark
ORDER BY g.grade DESC, s.name, s.marks
```

———————————

List every match with the goals scored by each team as shown.

```
SELECT mdate, team1, SUM(CASE WHEN teamid=team1 THEN 1 ELSE 0 END) AS score1,
  team2, SUM(CASE WHEN teamid = team2 THEN 1 ELSE 0 END) AS score2
  FROM game LEFT JOIN goal ON (id = matchid)
  GROUP BY mdate, team1, team2
  ORDER BY mdate, matchid, team1, team2
```

———————————

Joining three tables

```
SELECT * FROM
   movie JOIN casting ON movie.id=movieid
         JOIN actor   ON actorid=actor.id
  WHERE actor.name='John Hurt'
```

———————————

Obtain a list, in alphabetical order, of actors who've had at least 30 starring roles.

```
SELECT name FROM casting JOIN actor ON actorid = actor.id
WHERE ord=1 GROUP BY name HAVING COUNT(name)>=30 ORDER BY name
```

———————————

先group by 后order by，order之中一个逆序一个顺序

```
SELECT title, COUNT(actorid) FROM casting JOIN actor ON actorid = actor.id JOIN movie ON
movieid = movie.id
WHERE yr = 1978
GROUP BY title
```

ORDER BY COUNT(actorid) DESC, title

———————————

IS NULL

SELECT name FROM teacher WHERE dept IS NULL

———————————

LEFT JOIN — all teachers will be selected (even some don't appear in dept)

```
SELECT teacher.name, dept.name
 FROM teacher LEFT JOIN dept
         ON (teacher.dept=dept.id)
```

———————————

RIGHT JOIN — all dept will be selected, even some don't appear in teacher

```
SELECT teacher.name, dept.name
 FROM teacher RIGHT JOIN dept
         ON (teacher.dept=dept.id)
```

———————————

COALESCE takes any number of arguments and returns the first value that is not null.

COALESCE(x,y,z) = x if x is not NULL
COALESCE(x,y,z) = y if x is NULL and y is not NULL
COALESCE(x,y,z) = z if x and y are NULL but z is not NULL
COALESCE(x,y,z) = NULL if x and y and z are all NULL

———————————

COALESCE can be useful when you want to replace a NULL value with some other value. In this example you show the name of the party for each MSP that has a party. For the MSP with no party (such as Canavan, Dennis) you show the string None.

```
SELECT name, party
     ,COALESCE(party,'None') AS aff
  FROM msp WHERE name LIKE 'C%'
```

———————————

Use the COALESCE function and a LEFT JOIN to print the teacher **name** and department name. Use the string 'None' where there is no department.

```
SELECT teacher.name, COALESCE(dept.name , 'None')
FROM teacher LEFT JOIN dept ON teacher.dept = dept.id
```

———————————

Use COUNT and GROUP BY **dept.name** to show each department and the number of staff. Use a RIGHT JOIN to ensure that the Engineering department is listed.

```
SELECT dept.name, COUNT(teacher.name)
FROM teacher RIGHT JOIN dept ON teacher.dept = dept.id
GROUP BY dept.name
```

———————————

SELF JOIN

Execute the self join shown and observe that b.stop gives all the places you can get to from Craiglockhart, without changing routes. Change the query so that it shows the services from Craiglockhart to London Road.

```
SELECT a.company, a.num, a.stop, b.stop
FROM route a JOIN route b ON
```

(a.company=b.company AND a.num=b.num)
WHERE a.stop=53 AND b.stop = 149

———————————
More on SELF JOIN

SELECT a.company, a.num, stopa.name, stopb.name
FROM route a JOIN route b ON
(a.company=b.company AND a.num=b.num)
JOIN stops stopa ON (a.stop=stopa.id)
JOIN stops stopb ON (b.stop=stopb.id)
WHERE stopa.name='Craiglockhart' AND stopb.name = 'London Road'

———————————
SELF JOIN without using JOIN

SELECT DISTINCT a.company, a.num
FROM route a, route b
WHERE a.company = b.company AND a.num=b.num
AND a.stop = 115 AND b.stop = 137

———————————

求余数： MOD(被除数，除数)

SELECT DISTINCT city FROM station WHERE MOD(id, 2)=0

———————————

cross join (also named: Cartesian join) definition:
Table 1 has x rows, table 2 has y rows, each row in table 1 is joined to each row in table 2, giving you x*y total rows

———————————
UNION (It's like append)

String Manipulation

———————————
SELECT first_name, SUBSTR(first_name, 1, 3) FROM employees
substring of first_name, starting from the 1st char, total length = 3

———————————
SELECT UPPER(first_name) FROM employees
SELECT LOWER(first_name) FROM employees
SELECT UCASE(first_name) FROM employees
upper and ucase are the same: change char to upper case
lower: change to lower case
———————————
Formatting date and time:
SELECT Birthdate, STRFTIME('%Y', Birthdate) AS Year,
STRFTIME('%m', Birthdate) AS Month,

STRFTIME('%d', Birthdate) AS Day
FROM employees
————————————
get current date:
SELECT DATE('now')
SELECT STRFTIME('%Y %m %d', 'now')
SELECT Birthdate, DATE(('now') - Birthdate) AS Age FROM Employees

select employees who have been hired for more than 15 years
SELECT LastName, HireDate
FROM Employees
WHERE STRFTIME('%Y', DATE('now')-HireDate)>=15
ORDER BY LastName ASC

Data Profiling:
Looking at descriptive statistics or object data information — examining data for completeness and accuracy; important to understand data before you query it.

Object data profile:
number of rows, table size, when the object was last updated

Column data profiling:
column data type, number of distinct values (missing data?), number of rows with NULL values, descriptive stats(min, max, avg, st.dev.)

round up to the closest integer:
SELECT CEIL(salary) FROM employees

round down: FLOOR()

REPLACE(variable_name, 'char to replace', 'replace to what char')
e.g. remove all '0's in an integer:
SELECT REPLACE(salary, '0', '') FROM employees

选择最大的数字并且dataset里有多少并列最大的数.

SELECT (salary*months) AS earnings, COUNT(*) FROM employee
GROUP BY earnings ORDER BY earnings DESC LIMIT 1

TRUNCATE(n, d): truncate a float number n to d decimals