

Database Assignment 1

Name : Yujan Basnet

Roll No: 12

Que1: Japanese Cities' Names

SQL Script Solution:

```
SELECT NAME
FROM CITY
WHERE COUNTRYCODE = 'JPN';
```

Screenshot:

The screenshot shows the HackerRank interface for the 'Japanese Cities' Names problem. On the left, a sidebar contains links for Problem, Submissions, Leaderboard, and Discussions. The main problem area on the left states: 'Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN. The CITY table is described as follows:'. Below this is a table for the CITY table:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The right side of the interface shows the SQL editor with the following code:

```
1
2 */
3
4 Enter your query here and follow these instructions:
5 1. Please append a semicolon ";" at the end of the query and enter your query in a single line to avoid
6 error.
7
8 2. The AS keyword causes errors, so follow this convention: "Select t.Field From table1 t" instead of
9 "select t.Field From table1 AS t"
10 3. Type your code immediately after comment. Don't leave any blank line.
11
12 SELECT NAME
13 FROM CITY
14 WHERE COUNTRYCODE = 'JPN';
```

Below the editor, there are buttons for 'Run Code' and 'Submit Code'. A 'Congratulations!' message states: 'You have passed the sample test cases. Click the submit button to run your code against all the test cases.' Below this, a 'Sample Test case 0' section shows the 'Your Output (stdout)' as:

```
1 Heyagawa
2 Ageo
3 Sayama
4 Omuta
5 Tokuyama
```

Que2: Weather Observation Station 3

SQL Script Solution:

```
SELECT DISTINCT(CITY) FROM STATION
WHERE (MOD(ID,2)) = 0;
```

Screenshot:

The screenshot displays the HackerRank interface for the 'Weather Observation Station 3' challenge. On the left, the 'Problem' section explains the task: to query a list of city names from the 'STATION' table for cities with an even ID, excluding duplicates. It also provides the table schema for 'STATION'.

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

Below the schema, it notes that 'LAT_N' is the northern latitude and 'LONG_W' is the western longitude.

The right side of the screenshot shows the SQL editor with the solution: `SELECT DISTINCT(CITY) FROM STATION WHERE (MOD(ID,2)) = 0;`. Below the editor are buttons for 'Run Code' and 'Submit Code'. A notification indicates that the user has earned 10.00 points and is 25% of the way to a star badge. A large green 'Congratulations' banner states, 'You solved this challenge. Would you like to challenge your friends?' with a 'Next Challenge' button. At the bottom, a 'Test case 0' status is shown.

Que3: Weather Observation Station 5

SQL Script Solution:

```
-- SMALLEST
SELECT CITY,LENGTH(CITY) FROM STATION
ORDER BY LENGTH(CITY),CITY
LIMIT 1;
```

```
-- LARGEST
SELECT CITY, LENGTH(CITY) FROM STATION
ORDER BY LENGTH(CITY) DESC,CITY
LIMIT 1;
```

Screenshot:

The screenshot shows the HackerRank interface for the 'Weather Observation Station 5' challenge. On the left, the problem description asks to query the two cities in the **STATION** table with the shortest and longest city names, along with their lengths. It specifies that if there are ties, the city that comes first alphabetically should be chosen. Below the text is the **STATION** table schema:

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

Below the schema, it notes that **LAT_N** is the northern latitude and **LONG_W** is the western longitude. On the right, the SQL solution is entered in the editor:

```
-- SMALLEST
SELECT CITY,LENGTH(CITY) FROM STATION
ORDER BY LENGTH(CITY),CITY
LIMIT 1;

-- LARGEST
SELECT CITY, LENGTH(CITY) FROM STATION
ORDER BY LENGTH(CITY) DESC,CITY
LIMIT 1;
```

The interface shows the code is successful, with a 'Run Code' button and a 'Submit Code' button. A success message states: 'You have earned 30.00 points! You are now 30 points away from the 1st star for your sql badge.' The progress is 63% (50/80). A green banner at the bottom says 'Congratulations' and 'You solved this challenge. Would you like to challenge your friends?' with a 'Next Challenge' button.

Que4: The Blunder

SQL Script Solution:

```
SELECT CEILING(AVG(salary)-AVG(REPLACE(salary, 0,'')))  
FROM employees;
```

Screenshot:

The screenshot displays the HackerRank interface for the 'The Blunder' challenge. The left sidebar contains navigation links: Problem, Submissions, Leaderboard, and Discussions. The main content area on the left provides the problem description, input format, constraints, and sample input/output.

Problem Description: Samantha was tasked with calculating the average monthly salaries for all employees in the **EMPLOYEES** table, but did not realize her keyboard's **0** key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary. Write a query calculating the amount of error (i.e.: *actual – miscalculated* average monthly salaries), and round it up to the next integer.

Input Format

The **EMPLOYEES** table is described as follows:

Column	Type
ID	Integer
Name	String
Salary	Integer

Note: Salary is per month.

Constraints

$1000 < \text{Salary} < 10^5$.

Sample Input

ID	Name	Salary
1	Kristeen	1420
2	Ashley	2006
3	Julia	2210
4	Maria	3000

Sample Output

2861

The right side of the interface shows the SQL solution and test case results. The SQL query is:

```
1 SELECT CEILING(AVG(salary)-AVG(REPLACE(salary, 0,'')))  
2 FROM employees;  
3
```

The test case results show a 'Success' message and the expected output of 2253.

Que5: Weather Observation Station 18

SQL Script Solution:

```
SELECT ROUND( ABS(max(Lat_N) - min(Lat_N)) + ABS(max(LONG_W) - min(LONG_W)), 4) AS Manhattan_Distance
FROM STATION;
```

Screenshot:

The screenshot displays the HackerRank interface for the 'Weather Observation Station 18' challenge. The left sidebar contains navigation links: Problem, Submissions, Leaderboard, and Discussions. The main content area on the left provides the problem details:

Consider $P_1(a, b)$ and $P_2(c, d)$ to be two points on a 2D plane.

- a happens to equal the minimum value in Northern Latitude (LAT_N in STATION).
- b happens to equal the minimum value in Western Longitude (LONG_W in STATION).
- c happens to equal the maximum value in Northern Latitude (LAT_N in STATION).
- d happens to equal the maximum value in Western Longitude (LONG_W in STATION).

Query the [Manhattan Distance](#) between points P_1 and P_2 and round it to a scale of 4 decimal places.

Input Format

The STATION table is described as follows:

STATION	
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

The right side of the screenshot shows the SQL editor with the following query:

```
1 SELECT ROUND( ABS(max(Lat_N) - min(Lat_N)) + ABS(max(LONG_W) - min(LONG_W)), 4) AS Manhattan_Distance
2 FROM STATION;
```

Below the editor, there are buttons for 'Run Code' and 'Submit Code'. A notification states: 'You have earned 25.00 points! You are now 85 points away from the 2nd star for your sql badge.' The progress bar shows 11% completion (90/175 points).

A green 'Congratulations' banner reads: 'You solved this challenge. Would you like to challenge your friends?' with social media icons and a 'Next Challenge' button.

The 'Test case 0' section shows the compiler message 'Success', the input (stdin) as 'INPUT', and the expected output as '259.6859'.

Que6: Average Population of Each Continent

SQL Script Solution:

```
SELECT COUNTRY.CONTINENT, FLOOR(AVG(CITY.POPULATION)) AS  
AVERAGE_POPULATION  
FROM COUNTRY  
INNER JOIN CITY ON COUNTRY.Code = CITY.CountryCode  
GROUP BY COUNTRY.CONTINENT;
```

Screenshot:

The screenshot displays the HackerRank interface for the SQL challenge 'Average Population of Each Continent'. The problem description on the left states: 'Given the CITY and COUNTRY tables, query the names of all the continents (COUNTRY.Continent) and their respective average city populations (CITY.Population) rounded down to the nearest integer. Note: CITY.CountryCode and COUNTRY.Code are matching key columns. Input Format: The CITY and COUNTRY tables are described as follows:'

The CITY table structure is as follows:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

The COUNTRY table structure is as follows:

Field	Type
CODE	VARCHAR2(3)
NAME	VARCHAR2(44)
CONTINENT	VARCHAR2(13)
REGION	VARCHAR2(25)
SURFACEAREA	NUMBER
INDEPYEAR	VARCHAR2(5)
POPULATION	NUMBER
LIFEEXPECTANCY	VARCHAR2(4)
GNP	NUMBER
GNPOLD	VARCHAR2(9)
LOCALNAME	VARCHAR2(44)
GOVERNMENTFORM	VARCHAR2(44)

The SQL solution provided in the editor is:

```
1 SELECT COUNTRY.CONTINENT, FLOOR(AVG(CITY.POPULATION)) AS  
2 AVERAGE_POPULATION  
3 FROM COUNTRY  
4 INNER JOIN CITY ON COUNTRY.Code = CITY.CountryCode  
5 GROUP BY COUNTRY.CONTINENT;
```

The test results show a 'Success' message with the following expected output:

Expected Output
1 Oceania 109189
2 South America 147435

Que7: The PADS

SQL Script Solution:

```
SELECT
    CONCAT(name, '(', LEFT(occupation, 1), ')')
FROM occupations
ORDER BY name;
```

```
SELECT
CONCAT(
    "There are a total of ",
    COUNT(occupation),
    " ",
    LOWER(occupation),
    "s.")
FROM occupations
GROUP BY occupation
ORDER BY count(occupation), occupation;
```

Screenshot:

The screenshot shows the HackerRank interface for the 'The PADS' problem. The left sidebar contains navigation links: Problem, Submissions, Leaderboard, and Discussions. The main content area is divided into two columns. The left column contains the problem description, which asks for two SQL queries: one to list names with their first profession in parentheses, and another to count the occurrences of each profession and format the output. It also includes an 'Input Format' table and a 'Sample Input' table. The right column shows the SQL solution code, which is a MySQL script. Below the code, there is a 'Congratulations!' message and a list of sample test cases. The test cases show the output of the SQL queries for a given set of input data.

Problem

Generate the following two result sets:

- Query an alphabetically ordered list of all names in **OCCUPATIONS**, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example:
`ASingerName(S)`.
- Query the number of occurrences of each occupation in **OCCUPATIONS**. Sort the occurrences in ascending order, and output them in the following format:

There are a total of [occupation_count] [occupation]s.

where [occupation_count] is the number of occurrences of an occupation in **OCCUPATIONS** and [occupation] is the lowercase occupation name. If more than one Occupation has the same [occupation_count], they should be ordered alphabetically.

Note: There will be at least two entries in the table for each type of occupation.

Input Format

Column	Type
Name	String
Occupation	String

The **OCCUPATIONS** table is described as follows:
Occupation will only contain one of the following values: **Doctor, Professor, Singer** or **Actor**.

Sample Input

An **OCCUPATIONS** table that contains the following records:

Name	Occupation
Samantha	Doctor
Julia	Actor
Maria	Actor
Meera	Singer
Ashely	Professor
Ketty	Professor

SQL Solution

```
1 SELECT
2     CONCAT(name, '(', LEFT(occupation, 1), ')')
3 FROM occupations
4 ORDER BY name;
5
6 SELECT
7     CONCAT(
8         "There are a total of ",
9         COUNT(occupation),
10        " ",
11        LOWER(occupation),
12        "s.")
13 FROM occupations
14 GROUP BY occupation
15 ORDER BY count(occupation), occupation;
```

Line: 6 Col: 7

[Upload Code as File](#) [Run Code](#) [Submit Code](#)

Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

10	Julia(D)
11	Ketty(A)
12	Kristeen(S)
13	Maria(P)
14	Meera(P)
15	Naomi(P)
16	Priya(D)
17	Priyanka(P)
18	Samantha(A)
19	There are a total of 3 doctors.
20	There are a total of 4 actors.
21	There are a total of 4 singers.
22	There are a total of 7 professors.

Que8: Type of Triangle

SQL Script Solution:

```
SELECT
CASE
WHEN a + b <= c OR b + c <= a OR a + c <= b THEN 'Not A Triangle'
WHEN a=b AND b=c AND a=c THEN 'Equilateral'
WHEN a=b OR b=c OR a=c THEN 'Isosceles'
ELSE "Scalene"
END
FROM triangles;
```

Screenshot:

The screenshot shows the HackerRank interface for the 'Type of Triangle' problem. On the left, the problem description states: 'Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table: Equilateral: It's a triangle with 3 sides of equal length. Isosceles: It's a triangle with 2 sides of equal length. Scalene: It's a triangle with 3 sides of differing lengths. Not A Triangle: The given values of A, B, and C don't form a triangle.' Below this, the 'Input Format' table is shown with columns A, B, and C, all of type Integer. A sample input table is provided with 5 rows of side lengths. The 'Sample Output' shows the corresponding triangle types: Isosceles, Equilateral, Scalene, Not A Triangle, and Scalene. The main area displays the SQL query solution, which is a CASE statement. The query is: SELECT CASE WHEN a + b <= c OR b + c <= a OR a + c <= b THEN 'Not A Triangle' WHEN a=b AND b=c AND a=c THEN 'Equilateral' WHEN a=b OR b=c OR a=c THEN 'Isosceles' ELSE 'Scalene' END FROM triangles;. The interface shows 'Congratulations!' and 'You have passed the sample test cases.' Below this, a list of test cases is shown, all of which passed.

Problem

Write a query identifying the type of each record in the **TRIANGLES** table using its three side lengths. Output one of the following statements for each record in the table:

- **Equilateral:** It's a triangle with 3 sides of equal length.
- **Isosceles:** It's a triangle with 2 sides of equal length.
- **Scalene:** It's a triangle with 3 sides of differing lengths.
- **Not A Triangle:** The given values of A, B, and C don't form a triangle.

Input Format

The **TRIANGLES** table is described as follows:

Column	Type
A	Integer
B	Integer
C	Integer

Each row in the table denotes the lengths of each of a triangle's three sides.

Sample Input

A	B	C
20	20	23
20	20	20
20	21	22
13	14	30

Sample Output

```
Isosceles
Equilateral
Scalene
Not A Triangle
```

Explanation

```
1 SELECT
2 CASE
3   WHEN a + b <= c OR b + c <= a OR a + c <= b THEN 'Not A Triangle'
4   WHEN a=b AND b=c AND a=c THEN 'Equilateral'
5   WHEN a=b OR b=c OR a=c THEN 'Isosceles'
6   ELSE "Scalene"
7   END
8 FROM triangles;
```

Line: 1 Col: 1

Upload Code as File

Run Code Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Test Case	Result
1	Isosceles
2	Equilateral
3	Isosceles
4	Equilateral
5	Scalene
6	Not A Triangle
7	Scalene
8	Scalene
9	Scalene
10	Scalene
11	Scalene
12	Not A Triangle
13	Not A Triangle
14	Scalene
15	Equilateral

Que9: Weather Observation Station 13

SQL Script Solution:

```
SELECT TRUNCATE(SUM(lat_n), 4)
FROM station
WHERE lat_n > 38.7880 AND lat_n < 137.2345;
```

Screenshot:

The screenshot displays the HackerRank interface for the challenge 'Weather Observation Station 13'. The problem description asks for the sum of Northern Latitudes (LAT_N) from the STATION table, truncated to 4 decimal places, for values between 38.7880 and 137.2345. The STATION table schema is provided with fields ID, CITY, STATE, LAT_N, and LONG_W. The SQL solution is shown in the editor, and the test case results show a successful compilation and execution, resulting in the expected output 36354.8135.

Problem

Query the sum of Northern Latitudes (LAT_N) from **STATION** having values greater than 38.7880 and less than 137.2345. Truncate your answer to 4 decimal places.

Input Format

The **STATION** table is described as follows:

STATION	
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

SQL Solution

```
1 SELECT TRUNCATE(SUM(lat_n), 4)
2 FROM station
3 WHERE lat_n > 38.7880 AND lat_n < 137.2345;
4
```

Test Case Results

Test case 0: Success

Input (stdin): INPUT

Expected Output: 36354.8135

Congratulations

You have earned 10.00 points! You are now 15 points away from the 2nd star for your sql badge. 84% 160/175

Next Challenge

Que10: The Report

SQL Script Solution:

```
SELECT
CASE
WHEN s.marks <70 THEN null
ELSE s.name
END,
(SELECT
grade
FROM grades WHERE s.marks >= min_mark AND s.marks <= max_mark) AS grade,
s.marks
FROM students s
ORDER BY grade DESC, s.name ASC, s.marks ASC;
```

Screenshot:

Problem

ID	Name	Marks	Grade
4	Scarlet	78	
5	Ashley	63	
6	Jane	81	

Sample Output

```
Maria 10 99
Jane 9 81
Julia 9 88
Scarlet 8 78
NULL 7 63
NULL 7 68
```

Note

Print "NULL" as the name if the grade is less than 8.

Explanation

Consider the following table with the grades assigned to the students:

ID	Name	Marks	Grade
1	Julia	88	9
2	Samantha	68	7
3	Maria	99	10
4	Scarlet	78	8
5	Ashley	63	7
6	Jane	81	9

So, the following students got 8, 9 or 10 grades:

- Maria (grade 10)
- Jane (grade 9)
- Julia (grade 9)
- Scarlet (grade 8)

SQL Editor

```
1 SELECT
2 CASE
3 WHEN s.marks <70 THEN null
4 ELSE s.name
5 END,
6 (SELECT
7 grade
8 FROM grades WHERE s.marks >= min_mark AND s.marks <= max_mark) AS grade,
9 s.marks
10 FROM students s
11 ORDER BY grade DESC, s.name ASC, s.marks ASC;
```

Compiler Message

Success

Input (stdin)

```
1 INPUT
```

Expected Output

```
1 Britney 10 95
```

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

Next Challenge