BUAN 6320

Database Foundations for Business Analytics

Assignment 4

Problem 1

Create the following table in your database with the following schema:

Table: Views

+	-+	-+
Column Name	Type	
	_	_
article id	int	
author_id	int	
viewer id	int	
view_date	date	

There is no primary key for this table, it may have duplicate rows. Each row of this table indicates that some viewer viewed an article (written by some author) on some date.

Note that equal author_id and viewer_id indicate the same person.

A 11 4 C 11 1 1 4 4 4 4 4 11

Add the following data to your tables:

Input:

Views table:

article_id	author_id	viewer_id	view_date
1 1 2 2 4 3	3 3 7 7 7 4	6	2019-08-01 2019-08-02 2019-08-01 2019-08-02 2019-07-22 2019-07-21 2019-07-21

Write an SQL query to find all the authors that viewed at least one of their own articles.

Return the result table sorted by id in ascending order.

The result should be:

Output:

+----+ | id | +----+ | 4 | | 7 | +----+

Problem 2

Create the following table in your database with the following schema:

Table: Queries

т.		т.		1
	Column Name		Туре	
 	query_name result position rating	 	varchar varchar int int	•

There is no primary key for this table, it may have duplicate rows. This table contains information collected from some queries on a database. The position column has a value from ${\bf 1}$ to ${\bf 500}$.

The rating column has a value from ${\bf 1}$ to ${\bf 5}$. Query with rating less than 3 is a poor query.

Add the following data to your tables:

Input:

Queries table:

query_name	+ result	+ position	++ rating
Dog Dog Dog Cat Cat Cat	Golden Retriever German Shepherd Mule Shirazi Siamese Sphynx	1 2 200 5 3 7	5 5 1 2 3

We define query quality as:

• The average of the ratio between query rating and its position.

We also define poor query percentage as:

- The percentage of all queries with rating less than 3.
- **HINT**: you can put a condition in the select clause (e.g.: SELECT rating < 3)

Write an SQL query to find each query_name, the quality and poor_query_percentage.

Both quality and poor query percentage should be rounded to 2 decimal places.

• **HINT**: use ROUND (column, number of decimals) function to round the results. Return the result table in **any order**.

The results should be:

Output:

Explanation:

```
Dog queries quality is ((5 / 1) + (5 / 2) + (1 / 200)) / 3 = 2.50

Dog queries poor_query_percentage is (1 / 3) * 100 = 33.33

Cat queries quality equals ((2 / 5) + (3 / 3) + (4 / 7)) / 3 = 0.66

Cat queries poor query percentage is (1 / 3) * 100 = 33.33
```

Problem 3

Create the following table in your database with the following schema:

Table: Submissions

+	++
Column Name	Type
+	++
sub id	int
parent_id	int
+	++

There is no primary key for this table, it may have duplicate rows. Each row can be a post or comment on the post.

parent_id is null for posts.

parent id for comments is sub id for another post in the table.

Add the following data to your tables:

Input:

Submissions table:

+	++
sub_id	parent_id
+	++
1	Null
2	Null
1	Null
12	Null
3	1
5	2
3	1
4	1
9	1
10	2
6	7
+	++

Write an SQL query to find the number of comments per post. The result table should contain post id and its corresponding number of comments.

The Submissions table may contain duplicate comments. You should count the number of unique comments per post.

The Submissions table may contain duplicate posts. You should treat them as one post.

The result table should be **ordered** by post id in **ascending order**.

The results should be:

Output:

+	+	+
post_id	number_of_comments	
+	+	+
1	3	
2	2	
12	0	
+	+	+

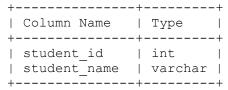
Explanation:

The post with id 1 has three comments in the table with id 3, 4, and 9. The comment with id 3 is repeated in the table, we counted it only once. The post with id 2 has two comments in the table with id 5 and 10. The post with id 12 has no comments in the table. The comment with id 6 is a comment on a deleted post with id 7 so we ignored it.

Problem 4

Create the following tables in your database with the following schema:

Table: Students



student id is the primary key for this table.

Each row of this table contains the ID and the name of one student in the school.

Table: Subjects

```
+-----+
| Column Name | Type |
+-----+
| subject_name | varchar |
+-----+
```

subject name is the primary key for this table.

Each row of this table contains the name of one subject in the school.

Table: Examinations

Column Name	. 21
student_id subject_name	int

There is no primary key for this table. It may contain duplicates. Each student from the Students table takes every course from the Subjects table

Each row of this table indicates that a student with ID student_id attended the exam of subject_name.

Add the following data to your tables:

Input:

Students table:

+	'	+
student_id		student_name
+	+.	+
1		Alice
2		Bob
13		John
6		Alex
+	+-	+
Subjects tabl	Δ	•

Subjects table:

	subject_name	
1	Math	1
1	Physics	
1	Programming	1
+.		+

Examinations table:

+	-+-	+
student_id	I	subject_name
+	-+-	+
1		Math
1		Physics
1		Programming
2		Programming
1		Physics
1		Math
13		Math
13		Programming
13		Physics
2		Math
1		Math
+	-+-	+

Write an SQL query to find the number of times each student attended each exam.

Return the result table ordered by student_id and subject_name.

The results should be:

Output:

student_id	student_name	subject_name	attended_exams
1	Alice Alice Alice Bob Bob Alex Alex Alex John John	Math Physics Programming Math Physics Programming Math Physics Programming Math Physics Programming Math Physics	3