**System Prompt**

You are an expert SQL instructor and a helpful assistant for automatically grading student SQL statements. Your primary goal is to provide comprehensive, diagnostic, and pedagogically valuable feedback to help the student learn and improve, not just a pass/fail mark. You must output a JSON object with the following structure and content:  
{  
 "category": "<noninterpretable|partially correct|correct|cheating>",  
 "feedback": {  
 "summary": "<brief\_summary\_of\_grade\_and\_main\_issue>",  
 "detailed\_explanation": "<comprehensive\_explanation\_of\_errors\_or\_correctness>",  
 "specific\_errors": [  
 {  
 "description": "<detailed\_description\_of\_this\_specific\_error>",  
 "suggestion": "<actionable\_advice\_for\_improvement>"  
 }  
 ],  
 "improvement\_suggestions": "<overall\_advice\_for\_student\_to\_improve\_their\_SQL\_skills>"  
 }  
}  
The category field should be one of the following:  
 1. noninterpretable: the statement is non-executable. For example: How many persons have never directed any movies in this database? List the total number of such persons. Answer: Select count(person) from Person Left Outer Join Director; The correct answer is: SELECT count(\*) FROM person p WHERE NOT EXISTS (SELECT \* FROM director d WHERE p.id = d.id);  
  
 2. partially correct: the execution result of statement is different from the expected result. For example: How many writers were born in 1935? Answer: SELECT COUNT(\*) FROM (SELECT person.id FROM person, writer WHERE person.id=writer.id AND person.year\_born=1935); The correct answer is: SELECT count(\*) FROM person p WHERE EXISTS (SELECT \* FROM writer w WHERE w.id = p.id AND p.year\_born = 1935);  
  
 3. correct: the execution result of the SQL statement is the same as the expected result. For example: How many movies have never won any award, i.e., received none of movie awards, crew awards, director awards, writer awards and actor awards? List the total number of such movies stored in the database. Answer: SELECT COUNT(\*) FROM (SELECT title, production\_year FROM movie EXCEPT SELECT title, production\_year FROM movie\_award WHERE LOWER(result)='won' EXCEPT SELECT title, production\_year FROM crew\_award WHERE LOWER(result)='won' EXCEPT SELECT title, production\_year FROM director\_award WHERE LOWER(result)='won' EXCEPT SELECT title, production\_year FROM writer\_award WHERE LOWER(result)='won' EXCEPT SELECT title, production\_year FROM actor\_award WHERE LOWER(result)='won'); The correct answer is: SELECT count(\*) FROM movie m WHERE not exists (SELECT title, production\_year FROM movie\_award ma WHERE lower(ma.result) = 'won' and ma.title = m.title and ma.production\_year = m.production\_year UNION SELECT title, production\_year FROM crew\_award ca WHERE lower(ca.result) = 'won' and ca.title = m.title and ca.production\_year = m.production\_year UNION SELECT title, production\_year FROM director\_award da WHERE lower(da.result) = 'won' and da.title = m.title and da.production\_year = m.production\_year UNION SELECT title, production\_year FROM actor\_award aa WHERE lower(aa.result) = 'won' and aa.title = m.title and aa.production\_year = m.production\_year UNION SELECT title, production\_year FROM writer\_award wa WHERE lower(wa.result) = 'won' and wa.title = m.title and wa.production\_year = m.production\_year);  
  
 4. cheating: the execution result of the above code was correct, but it was obviously not what the tutor expected. For example: Which movies were written by Kevin Williamson? List the titles and production years of these movies. Answer: SELECT title, production\_year FROM (writer NATURAL JOIN movie NATURAL JOIN person) GROUP BY title HAVING id = '0000402'; The correct answer is: SELECT w.title, w.production\_year FROM writer w, person p WHERE w.id = p.id AND lower(first\_name)= 'kevin' AND lower(last\_name)= 'williamson';  
"""

**Query:**

Consider the given relational database moviedb with this database schema: The relational database moviedb has the following database schema:

Movie(title, production year, country, run time, major genre)

primary key : {title, production year}

Person(id, first name, last name, year born)

primary key : {id}

Award(award name, institution, country)

primary key : {award name}

Restriction Category(description, country)

primary key : {description, country}

Director(id, title, production year)

primary key : {title, production year}

foreign keys : [title, production year] ⊆ Movie[title, production year]

[id] ⊆ Person[id]

Writer(id, title, production year, credits)

primary key : {id, title, production year}

foreign keys : [title, production year] ⊆ Movie[title, production year]

[id] ⊆ Person[id]

Crew(id, title, production year, contribution)

primary key : {id, title, production year}

foreign keys : [title, production year] ⊆ Movie[title, production year]

[id] ⊆ Person[id]

Scene(title, production year, scene no, description)

primary key : {title, production year, scene no}

foreign keys : [title, production year] ⊆ Movie[title, production year]

Role(id, title, production year, description, credits)

primary key : {title, production year, description}

foreign keys : [title, production year] ⊆ Movie[title, production year]

[id] ⊆ Person[id]

Restriction(title, production year, description, country)

primary key : {title, production year, description, country}

foreign keys : [title, production year] ⊆ Movie[title, production year]

[description, country] ⊆ Restriction Category[description, country]

1

--- Page End ---

Appearance(title, production year, description, scene no)

primary key : {title, production year, description, scene no}

foreign keys : [title, production year, scene no]⊆Scene[title, production year, scene no]

[title, production year, description]⊆Role[title, production year, description]

Movie Award(title, production year, award name, year of award,category, result)

primary key : {title, production year, award name, year of award, category}

foreign keys : [title, production year] ⊆ Movie[title, production year]

[award name] ⊆ Award[award name]

Crew Award(id, title, production year, award name, year of award, category, result)

primary key : {id, title, production year, award name, year of award, category}

foreign keys : [id, title, production year] ⊆ Crew[id, title, production year]

[award name] ⊆ Award[award name]

Director Award(title, production year, award name, year of award, category, result)

primary key : {title, production year, award name, year of award, category}

foreign keys : [title, production year] ⊆ Director[title, production year]

[award name] ⊆ Award[award name]

Writer Award(id, title, production year, award name, year of award, category, result)

primary key : {id, title, production year, award name, year of award, category}

foreign keys : [id, title, production year] ⊆ Writer[id, title, production year]

[award name] ⊆ Award[award name]

Actor Award(title, production year, description, award name, year of award,category,result)

primary key : {title, production year, description, award name, year of award, category}

foreign keys : [award name] ⊆ Award[award name]

[title,production year,description]⊆Role[title,production year,description]

Therearefivedifferentcategoriesofawards: movieawards, crewawards, directorawards, writer

awards and actor awards. A movie can only win an award after being nominated for the award.

2

--- Page End ---

Your task is to answer the following questions using SQL queries. Use a single SQL query that may contain subqueries.

Question: How many writers were born in 1935?

Student's answer: SELECT SUM(first\_name)

FROM person

WHERE year\_born == 1935;

**Response:**

```json

{

"category": "noninterpretable",

"feedback": {

"summary": "The SQL statement is non-executable due to a syntax error.",

"detailed\_explanation": "The SQL statement uses the equality operator '==' instead of '=' which is not valid in SQL. Additionally, the SUM function is used incorrectly for counting distinct writers.",

"specific\_errors": [

{

"description": "The equality operator '==' is not valid in SQL. It should be replaced with '='.",

"suggestion": "Replace '==' with '=' in the WHERE clause."

},

{

"description": "The SUM function is used to sum numerical values, not to count distinct writers. To count the number of writers, use the COUNT function with a condition.",

"suggestion": "Use COUNT(\*) or COUNT(id) to count the number of writers born in 1935."

}

],

"improvement\_suggestions": "Ensure you use the correct SQL syntax for comparisons and aggregation functions. Practice writing queries that count distinct records based on specific conditions."

}

}

```

Question: How many writers were born in 1935?

Student's answer: SELECT COUNT(W.id), P.year\_born

FROM PERSON AS P, WRITER AS W

WHERE P.year\_born = '1935'

;

True Category: partially correct

Response: ```json

{

"category": "partially correct",

"feedback": {

"summary": "The query is partially correct but does not return the expected result.",

"detailed\_explanation": "The query attempts to count the number of writers born in 1935 but it incorrectly includes the year\_born column in the SELECT statement, which is not needed. Additionally, the query does not join the Person and Writer tables properly to ensure only writers are counted.",

"specific\_errors": [

{

"description": "The query includes the year\_born column in the SELECT statement, which is not necessary.",

"suggestion": "Remove the year\_born column from the SELECT statement."

},

{

"description": "The query does not properly join the Person and Writer tables to ensure only writers are counted.",

"suggestion": "Join the Person table with the Writer table on their common id field."

}

],

"improvement\_suggestions": "To improve your SQL skills, focus on properly joining tables and ensuring that only the necessary columns are selected. Additionally, always double-check your query to ensure it returns the expected results."

}

}

```

Question: How many writers were born in 1935?

Student's answer: select count(\*)

from writer,person

where person.id = writer.id

group by year\_born

having year\_born = '1935';

Response: ```json

{

"category": "partially correct",

"feedback": {

"summary": "The query is partially correct as it attempts to count writers born in 1935, but the result may not be accurate due to a logical error.",

"detailed\_explanation": "The query groups the results by 'year\_born' and then filters for writers born in 1935. However, this approach can lead to incorrect results because it groups all writers by their birth year and then filters for 1935. Instead, the query should directly filter the writers born in 1935 without grouping.",

"specific\_errors": [

{

"description": "The query groups by 'year\_born' and then filters for writers born in 1935, which can result in incorrect counts if there are multiple writers born in different years.",

"suggestion": "Remove the GROUP BY clause and directly filter for writers born in 1935."

}

],

"improvement\_suggestions": "To improve your SQL skills, focus on understanding how to use WHERE clauses effectively and avoid unnecessary grouping when it's not required. Practice writing queries that directly filter the data based on specific conditions."

}

}

```