A Appendix

A.1 Prompt design and number of demonstrations for Spider

In Table 28, we analyze the performance of *Few-shot SQL-PaLM* method on different number of demonstrations (zero- vs. few-shot) and queries with different prompt designs. Overall, few-shot prompting outperforms zero-shot counterpart. We also explore the effect of different prompt design approaches on performance. For the LLM being queried (e.g. PaLM2), concise prompt is observed to be better. "Verbose" prompts are based on using natural language to describe database schema, which is closer to the way LLMs were trained, whereas "Concise" prompts use the symbols to describe the database schema, which has advantages of clearly presenting table structure. Examples of concise prompt and verbose prompt are provided in Appendix A.10.1 and A.10.2.

Prompt design	Adaptation setting	EX	TS
Concise	0-shot	81.2	76.0
Concise	4-shot	**82.7**	**77.3**
Verbose	0-shot	78.5	70.9
Verbose	4-shot	81.3	73.7

Table 28: Test-suite accuracy for different prompt design approaches in zero- and few-shot set-up on Spider Dev.

A.2 Prompt design for BIRD via few-shot prompting

In Table 29, we investigate various prompt design on BIRD datasets. Unlike Spider datasets (Table 28) where concise prompt works better, for BIRD datasets verbose prompt works superior.

Table 29: Execution accuracy of Text-to-SQL with different prompt designs across different SQL difficulty levels on BIRD datasets

	Simple	Moderate	Challenging	Total
Count	933	459	142	1534
Concise	49.95%	25.05%	19.01%	39.63%
Verbose	53.27%	29.85%	18.31%	43.02%

A.3 Descriptions of Robust Spider Datasets: Spider-SYN, Spider-Realistic, Spider-DK

Table 30: Information on different variants of Spider datasets with the purpose of evaluating robustness.

	Counts	Modification Category	Source	Modify Natural Question?	Modify Database Schema?	Add New Database Schema?	Examples
Spider-SYN	1034	Manually modifying natural language questions with synonym substitutions	Spider Dev.	Yes	No	No	Spider # Database Schema: concert_singer # stadium(Stadium_ID, Location, Name, Capacity, Highest, Lowest, Average) # singer(Singer_ID, Name, Country, Song_Name, Song_release_year, Age, Is_male) # concert(concert_ID, concert_Name, Theme, Stadium_ID, Year) # singer_in_concert(concert_ID, Singer_ID) # Q: How many singers do we have? Spider-SYN Q: How many vocalists do we have?
Spider-Realistic	508	Modify natural language questions to remove explicitly mentioning column names	Subset of Spider Dev	Yes	No	No	Spider # Database Schema: concert_singer Q: How many concerts are there in year 2014 or 2015? Q: How many concerts are there in 2014 or 2015? # No year
Spider-DK	535	Modify database schema to incorporate the domain knowledge	Subset of Spider Dev	Yes	Yes	Yes	# Database Schema: concert_singer Modify database column "Age" into "Birthday"; Replace its values from "52" to "1971-02-09 00:00:00" Q: List all song names by singers above the average age. # hard to answer "age"-related question

A.4 Tuning performance with different foundation models

Table 31 shows the results of tuning open-source models LLaMA7B, LLaMA13B, and LLaMA33B on Spider using the best input representation as reported in Gao et al. (2023a)[33]

Table 31: **Evaluations on Spider dev split using different foundation models.**

Foundation model	TS
LLaMA-7B	66.7%
LLaMA-2-CHAT-7B	69.6%
LLaMA-13B	68.6%
LLaMA-2-CHAT-13B	65.1%
LLaMA-33B	69.1%

A.5 Synthetic data

A.5.1 Synthetic data prompt design

Synthetic Data Prompt Design

You will be provided with a list of tables from a SQL database followed by a natural language query related to the database and the original SQL query answering the question. Your job is to understand the natural language queries and generate up to 3 different SQL queries using diverse commands from the original query while answering the question correctly. You need to make sure to use the same columns from the original query for the generated query. You will also generate a similarity score between the original and the generated query based on how closer they are syntactically.

Database tables schema are as follows:

```
CREATE TABLE customers (
  customer_id int,            -- unique customer id
  name varchar(100),          -- name of the customer
  email_address varchar(255), -- email address of the customer
);

CREATE TABLE order (
  order_id int,                -- unique order id.
  customer_id int,             -- unique customer id.
  order_amount decimal(10, 2), -- amount spent by the customer on the order
);
```

Question: Find the email of the top spending customer?

Original SQL query:

```
SELECT customers.first_name
FROM customers
JOIN order ON customers.customer_id = order.customer_id
GROUP BY customers.customer_id, customers.first_name
ORDER BY SUM(order.order_amount) DESC
LIMIT 1;
```

Output the generated queries and the similarity scores in a json list as follows:

---

[33] The numbers are taken from Gao et al. (2023a);

```
[
  {"sql":        // generated query-1,
   "similarity": // similarity score (0.0-1.0) for query-1
  },
  {...}
]
```

A.5.2 Synthetic data similarity score distribution

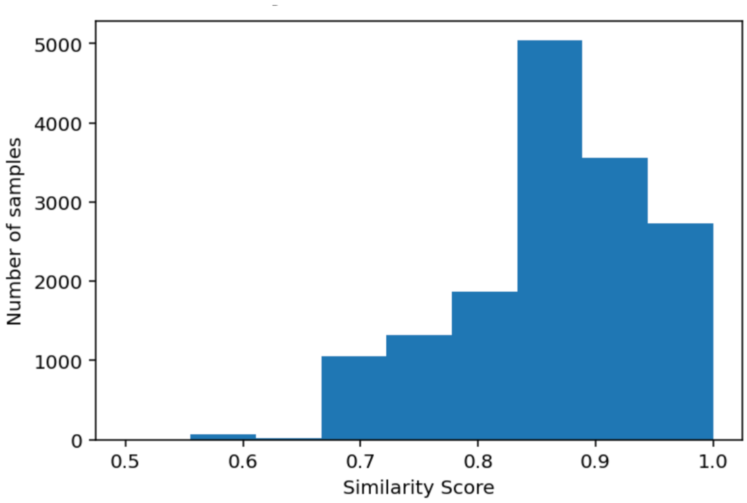

Figure 10: Histogram plot of synthetic data similarity scores

32 displays the statistics of the generated queries with correctness and similarity filters applied. It is important to note that lower similarity score indicates higher diversity.

Table 32: Synthetic data generation statistics for BIRD train split

	Correct SQL	Correct SQL + Similarity <= 0.9
Samples with 1+ generated queries	81.4%	78.8%

The mean, median, standard deviation of the similarity scores of all generated queries is reported in Table 33 and the histogram is plotted in Figure 10 in Appendix.

Table 33: Statistics of synthetic data similarity scores

Min	Max	Mean	Median	STD
0.5	1.0	0.85	0.85	0.07

A.6 Column Selection

A.6.1 Example of retrieval-based column selection

Template:

> Column name [column_name] of type [column_type] from the table [table_name]. Description: [column_description]. Value examples: [common_distinct_values].

Example:

Column name ‘size’ of type ‘STRING’ from the table ‘package’. Description: ‘package size dimensions’. Value examples: ‘small’, ‘medium’, ‘long’.

A.6.2 Program-aided column selection ablation studies

The accuracy of program-aided column selection is directly related to the accuracy of the preliminary SQL. The higher the accuracy of these preliminary SQL queries, the better the column selections based on them will be.

Table 34: Higher accuracy in preliminary SQL leads to better column selection

Accuracy (%)	**Table** (%)			**Column** (%)		
Preliminary SQL	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**F1**
43	92.96	91.15	92.04	84.75	86.62	85.67
50.2	94.10	94.41	94.25	87.97	90.87	89.40
55	93.76	95.12	94.44	89.62	91.69	90.64
58.8	94.64	96.14	95.39	90.66	92.60	91.62

A.6.3 Compare with other column selection methods

We presented other column selection methods: **LLM base**: prompting LLMs to request table and column selection (Prompt is in Sec. A.6.4). **LLM CoT**: Following Pourreza & Rafiei (2023), we add few-shot examples and use change-of-thought demonstrations to help the prompt (Prompt is in Sec. A.6.5). The model is PaLM-2 text-bison. **Automatic Annotation** (Lei et al., 2020b) is a proposes pattern matching approach. The last two lines are taken from Table 16 and Table 14 (Top 10), rounded by two decimals. The results in Table 35 indicates that program-aided algorithm outperform the other methods with a clear margin.

Table 35: Comparison of Table and Column selection

	Table selection			**Column selection**		
Methods	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**F1**
LLM baseline	0.71	0.88	0.75	0.24	0.83	0.35
LLM Few-shot	0.85	0.82	0.82	0.81	0.64	0.70
Automatic Annotation	0.87	0.74	0.80	0.68	0.90	0.78
Retrieval-based (Ours)	0.90	0.56	0.39	0.82	0.27	0.39
Program-aided (Ours)	**0.95**	**0.96**	**0.95**	**0.91**	**0.93**	**0.92**

A.6.4 LLM base prompt design of column selection

We start with a simple baseline which asks the model to select a schema in two steps. First, select tables and then columns. 1 Table selection

```
Select tables from my database named [database_name], to answer the given query.
   Tables:
    CREATE TABLE [table_name_1] ()
    CREATE TABLE [table_name_2] ()
    ...
   Query: '{query}'
   Only generate a list of comma separated table names without any spaces.
```

2 Column selection

```
Given a table definition and a natural language query,
I am interested in selecting columns related to the query:
   CREATE TABLE [table_name] (
     [column_name] [column_type],
   );
   Table Values:
   [3 row examples with header]
   The natural language query: {query}
   Select all related column names, including ids, from the table.
   Only generate a list of comma separated column names values without any spaces.
```

A.6.5 LLM Few-shot with CoT for column selection

Following Pourreza & Rafiei (2023), we add few-shot examples and use change-of-thought demonstrations to help the prompt.

```
You are an agent designed to find the schema_links  for generating SQL queries for each
question based on the database schema and Foreign keys.
Hint helps you to find the correct schema_links.
###
Few examples of this task are:
###
Schema of the database with sample rows and column descriptions:
#
CREATE TABLE users (
     user_id INT,
     ...
   );
   Table Values:
   User_id ...
   001 ...
Table users
  User_id: id of the user
  ...
Question: Among the lists created by user 4208563...
Answer: Let's think step by step. In the question , we are asked:
"user" so we need column = [lists_users.user_id]
"number of followers" so we need column = [lists.list_followers]...
Schema_links: [lists.list_followers,lists_users.user_subscriber,
lists.user_id = lists_user.user_id, lists.list_id = lists_user.list_id,
lists_users.user_id, 4208563, 1]

###
Schema of the database with sample rows and column descriptions:
#
CREATE TABLE [table_name] (
     [column_name] [column_type],
     ...
   );
   Table Values:
   [3 row examples with header]
Table [table_name]
  [column_name]: [column_description]
  ...
Question: [question]
```

```
Answer: Let’s think step by step.
```

A.7 Column and table data statistics of BIRD dataset

Table 36 illustrates the distribution of data regarding the number of columns and tables per example in the testing sets. The ground-truth distribution is also provided. Notably, BIRD presents a more challenging scenario compared to Spider, with an average of 73 total columns to be selected per example, of which only 3.7 columns are used in the ground truth. Similarly, the average number of tables is 7, and 1.9 tables are selected in the ground truth. However, it’s essential to acknowledge that these sets still deviate from real-world scenarios with thousands of columns. This direction should be further explored in the future.

Table 36: Statistics for table and columns in the data-sets

	BIRD Test						Spider Test					
	Mean	Min.	Max.	P75	P95	SDT	Mean	Min.	Max.	P75	P95	SDT
Tables	7	3	13	8	11.5	2.7	4	2	11	4	8.3	2.2
Columns	73	11	192	91.5	155.5	48.8	20.8	7	59	24	50	13.5
Columns per table	10.4	2	115	9	42.4	16.8	5.2	2	26	6	11.2	3.6
Ground-truth Tables	1.9	1	4	2	3	0.7	1.5	1	4	2	3	0.6
Ground-truth columns	3.7	1	9	5	6	1.4	2.4	1	6	3	5	1.2

Table 37: Average of number of columns and tables for questions for BIRD datasets: we compute the average number of table and columns for each questions.

	number of queries	Avg. number of table	Avg. of column counts	Median number of tables	Min. of column counts	Max. of column counts
BIRD-valid	1534	7.4	76.3	71	11	201
BIRD-train	9428	12.0	77.4	48	6	457

A.8 Exploration on combining submodules

An alternative approach involves combining different input configurations in previous sections into a single training experiment. This method entails integrating various elements, such as mixed training data, synthetic data, database content, and column selection, into the inputs for a single experiment. However, the outcomes of such experiments reveal that merging these components does not result in performance improvements over using them individually. This suggests that LLMs may struggle to effectively process and understand all the provided information simultaneously during tuning.

Type	**Train data**	**Method**	**Accuracy**
Tuning	BIRD + Spider	+ database content	58.80
Tuning	BIRD + Spider + Synthetic data	+ database content	58.35
Tuning	BIRD + Spider + Synthetic data	+ database content + soft column selection	58.08

Table 38: The effect of integrating all components into one training paradigm

A.9 Case Study of SQL Generation and Error Analysis

A.9.1 *SQL-PaLM* with few-shot Prompting

We present case studies of *Few-shot SQL-PaLM* in Table 39 and 40 for “correct” and “wrong” SQL generated by *Few-shot SQL-PaLM* based on test-suite accuracy of Spider dataset. Surprisingly, the majority of examples classified as "errors" by *Few-shot SQL-PaLM* were actually correct when evaluated by human experts, indicating the scores of *SQL-PaLM* might be significantly higher. The evaluation fails due to (1) ambiguous questions, exemplified by 1st example in Table 40, and (2) official test-suite evaluation struggles on evaluating creative solutions with output which deviate from that of the ground-truth. For instance, the 2nd example has multiple valid ground-truths; the 3rd example experiences type-related issues; the 4th example presents different formats (e.g. "full name" and "first name, second name" are equally semantically correct for the question. They both should be considered correct); and the 5th example is false negative due to the omission of the "distinct" keyword.

Regarding the "real" mistakes made by *Few-shot SQL-PaLM*, such as the sixth and seventh examples, we observed a departure from simple errors like syntax errors commonly found in other methods (Liu et al., 2023a). Instead, the mistakes made by *Few-shot SQL-PaLM* closely resemble those that human experts would make when developing the same solution, demonstrating its profound expertise in SQL. Another source of errors is the presence of a "confusing database schema," where *Few-shot SQL-PaLM* encounters difficulties in selecting the appropriate table or column when multiple equivalent options contain similar content (as illustrated in the 5th example of Table 40).

Tables 39 and 40 show the capabilities of *Few-shot SQL-PaLM*, demonstrating that it can efficiently handle complex SQL queries. It successfully deals with tasks such as joining multiple tables using various keywords (as observed in the 1st, 2nd, 4th, and 5th examples in Table 39 and all examples in Table 40), as well as employing nested SQL structures (as seen in the 3rd example of Table 39). Moreover, *Few-shot SQL-PaLM* exhibits the ability to generate creative and diverse SQL outputs that differ from the ground-truth but remain equally correct. This suggests a deep understanding of SQL content rather than mere memorization. Notable examples include the 3rd example in Table 39 and the 2nd, 3rd, 4th, and 5th examples in Table 40. Even in cases of errors, such as the 6th and 7th examples in Table 40, *Few-shot SQL-PaLM* presents alternative solutions distinct from the ground-truth. Furthermore, *Few-shot SQL-PaLM* demonstrates the ability to infer relevant SQL expression based on semantic meaning, i.e. "French singers" and "country=France," as well as "young to old" and "OrderBy age ASC" (as evident in the 1st and 2nd examples). This capability is attributed to the large-scale pretraining of LLMs.

A.9.2 Fine-tuned *SQL-PaLM*

Next, we present our manual investigation of the generated queries on the BIRD dev set. To quantify the error cases for the queries generated from the fine-tuned *Few-shot SQL-PaLM*, we randomly select 100 samples from BIRD dev set and categorize the queries that we considered mistakes based on the BIRD evaluation procedure.

Table 41 shows a breakdown of the different error types that we have identified and provided some examples in Table 42 for each category. We categorize the false positives as **Wrong Ground-Truth** (19%) which are examples from the dataset that do not correctly answer the original question. **Wrong Evidence** (5%) which denotes examples that have incorrect or misleading human-annotated evidences. **Ambiguous** (2%) which are examples where the question's meaning is ambiguous and open-ended (e.g., not clear what is expected from the question). **Evaluation Procedure** (5% denoting that the generated SQL query was in fact correct but was considered incorrect by the evaluation procedure (e.g., the generated queries contains additional selected columns). For a comprehensive list of examples from the different error types, refer to Table 42.

This investigation demonstrates that from 100 randomly sampled incorrect queries (out of a total of 584 for fine-tuned *Few-shot SQL-PaLM*), 31% of them are not correctly evaluated by the BIRD evaluation suite and if this sample size is representative of the full dev set, it shows that there is a performance upper bound of around 70%.

Table 39: *Few-shot SQL-PaLM* examples that are correct according to test-suite accuracy.

Data Schema	Question	Ground-truth	SQL-PaLM	Comment
Q1: **stadium**: Stadium_ID, Location, Name, Capacity, Highest, Lowest, Average **singer**: Singer_ID, Name, Country, Song_Name, Song_release_year, Age, Is_male **concert**: concert_ID, concert_Name, Theme, Stadium_ID, Year **singer_in_concert**: concert_ID, Singer_ID	What is the average, minimum, and maximum age of all singers from France?	`SELECT avg(age) , min(age) , max(age) FROM singer WHERE country = 'France'`	`SELECT avg(age) , min(age) , max(age) FROM singer WHERE country = "France"`	Inference on the relationship between "French" and "France"
	Show name, country, age for all singers ordered by age from the oldest to the youngest.	`SELECT name , country , age FROM singer ORDER BY age DESC`	`SELECT name , country , age FROM singer ORDER BY age DESC`	Inference based on understanding the age ranking via “order by age DESC”:
Q2: **city**: ID, Name, CountryCode, District, Population **sqlite_sequence**: name, seq **country**: Code, Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, Code2 **countrylanguage**: CountryCode, Language, IsOfficial, Percentage	What are the names of nations where both English and French are official languages?	`SELECT T1.Name FROM country AS T1 JOIN countrylanguage AS T2 ON T1.Code = T2.CountryCode WHERE T2.Language = "English" AND T2.IsOfficial = "T" INTERSECT SELECT T1.Name FROM country AS T1 JOIN countrylanguage AS T2 ON T1.Code = T2.CountryCode WHERE T2.Language = "French" AND T2.IsOfficial = "T"`	`SELECT T1.name FROM country AS T1 JOIN countrylanguage AS T2 ON T1.code = T2.countrycode WHERE T2.language = "English" AND T2.isofficial = "T" INTERSECT SELECT T1.name FROM country AS T1 JOIN countrylanguage AS T2 ON T1.code = T2.countrycode WHERE T2.language = "French" AND T2.isofficial = "T"`	Generate exact GT
Q3: **stadium**: Stadium_ID, Location, Name, Capacity, Highest, Lowest, Average **singer**: Singer_ID, Name, Country, Song_Name, Song_release_year, Age, Is_male **concert**: concert_ID, concert_Name, Theme, Stadium_ID, Year **singer_in_concert**: concert_ID,Singer_ID	What are the number of concerts that occurred in the stadium with the largest capacity ?	`select count(*) from concert where stadium_id = (select stadium_id from stadium order by capacity desc limit 1)`	`SELECT count(*) FROM concert AS T1 JOIN stadium AS T2 ON T1.stadium_id = T2.stadium_id WHERE T2.capacity = ( SELECT max(T3.capacity) FROM stadium AS T3 )`	Creative – different from ground-truth but accurate
Q4: **Student**: StuID, LName, Fname, Age, Sex, Major, Advisor, city_code **Has_Pet**: StuID, PetID **Pets**: PetID, PetType, pet_age, weight	Find the first name of students who have both cat and dog pets .	`select t1.fname from student as t1 join has_pet as t2 on t1.stuid = t2.stuid join pets as t3 on t3.petid = t2.petid where t3.pettype = 'cat' intersect select t1.fname from student as t1 join has_pet as t2 on t1.stuid = t2.stuid join pets as t3 on t3.petid = t2.petid where t3.pettype = 'dog'`	`SELECT T1.fname FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid WHERE T3.pettype = "cat" INTERSECT SELECT T1.fname FROM student AS T1 JOIN has_pet AS T2 ON T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid WHERE T3.pettype = "dog"`	Complex SQL containing multiple SQL keywords.
Q5: **continents**: ContId, Continent **countries**: CountryId, CountryName, Continent **car_makers**: Id, Maker, FullName, Country **model_list**: ModelId, Maker, Model **car_names**: MakeId, Model, Make **cars_data**: Id, MPG, Cylinders, Edispl, Horsepower, Weight, Accelerate, Year	What are the ids and names of all countries that either have more than 3 car makers or produce fiat model ?	`select t1.countryid , t1.countryname from countries as t1 join car_makers as t2 on t1.countryid = t2.country group by t1.countryid having count(*) > 3 union select t1.countryid , t1.countryname from countries as t1 join car_makers as t2 on t1.countryid = t2.country join model_list as t3 on t2.id = t3.maker where t3.model = 'fiat';`	`SELECT T1.countryid , T1.countryname FROM countries AS T1 JOIN car_makers AS T2 ON T1.countryid = T2.country GROUP BY T1.countryid HAVING count(*) > 3 UNION SELECT T1.countryid , T1.countryname FROM countries AS T1 JOIN car_makers AS T2 ON T1.countryid = T2.country JOIN model_list AS T3 ON T2.id = T3.maker WHERE T3.model = "fiat"`	Complex SQL containing multiple SQL keywords

Table 40: *Few-shot SQL-PaLM* outputs that are incorrect according to test suite evaluation. Indeed, according to human experts, the majority of these “error cases” are actually the correct SQL outputs.

Question 1:	**What is the maximum capacity and the average of all stadiums ? (id=16, database=concert_singer)**
Ground-truth:	`select max(capacity), average from stadium`
SQL-PALM:	`SELECT max(capacity) , avg(capacity) FROM stadium`
Comment:	Ambiguous question. The Ground-truth query would give the **average occupancy column** of each stadium, where as the SQL-PALM one would give the average of the **capacity column** of all stadiums. Without specific context, the SQL-PALM output matches the question better.
Human Expert	**True (Ground-truth issue)**
Question 2:	**Which airline has most number of flights? (id=229, database=flight_2)**
Ground-truth:	`SELECT T1.Airline FROM AIRLINES AS T1 JOIN FLIGHTS AS T2 ON T1.uid = T2.Airline GROUP BY T1.Airline ORDER BY count(*) DESC LIMIT 1`
SQL-PALM:	`SELECT T1.airline FROM airlines AS T1 JOIN flights AS T2 ON T1.uid = T2.airline GROUP BY T2.airline ORDER BY count(*) DESC LIMIT 1`
Comment:	Both `T1.Airline` (the `Airlines.Airline` column, which contains names of airlines) and `T2.airline` (the `flights.airline` column, which contains the `uid` of airlines) uniquely identify airlines, and grouping by either one gives the same results. The evaluation failed because the data contains 12 airlines each with 100 flights (i.e. every one is the maximum), and instability of the ordering produces different outcomes when both queries pick one of the maximums.
Human Expert	**True (Database and Eval issue)**
Question 3:	**What are the id, name and membership level of visitors who have spent the largest amount of money in total in all museum tickets? (id=420, database=museum_visit)**
Ground-truth:	`SELECT t2.visitor_id , t1.name , t1.Level_of_membership FROM visitor AS t1 JOIN visit AS t2 ON t1.id = t2.visitor_id GROUP BY t2.visitor_id ORDER BY sum(t2.Total_spent) DESC LIMIT 1`
Excution result:	(’3’, ’Arjen Robben’, 1), where ’3’ is a string
SQL-PALM:	`SELECT T1.id , T1.name , T1.level_of_membership FROM visitor AS T1 JOIN visit AS T2 ON T1.id = T2.visitor_id GROUP BY T1.id ORDER BY sum(T2.total_spent) DESC LIMIT 1`
Excution result:	(3, ’Arjen Robben’, 1), where 3 is an integer
Comment:	The result is actually correct. The highlighted differences `t2.visitor_id` and `t1.id` are equivalent because the two tables are joined on `t1.id = t2.visitor_id`. Evaluation failed because of the field type differences, as `t1.id` is `int` and `t2.visitor_id` is `text`.
Human Expert	**True (Database and Eval issue)**
Question 4:	**List the names of all winners who played in both 2013 and 2016. (id=447, database=wta_1)**
Ground-truth:	`SELECT winner_name FROM matches WHERE YEAR = 2013 INTERSECT SELECT winner_name FROM matches WHERE YEAR = 2016`
Excution result:	(’Angelique Kerber’,), (’Petra Kvitova’,) ...
SQL-PALM:	`SELECT T1.first_name , T1.last_name FROM players AS T1 JOIN matches AS T2 ON T1.player_id = T2.winner_id WHERE T2.year = 2013 INTERSECT SELECT T1.first_name , T1.last_name FROM players AS T1 JOIN matches AS T2 ON T1.player_id = T2.winner_id WHERE T2.year = 2016`
Execution result:	(’Angelique’, ’Kerber’), (’Petra’, ’Kvitova’) ...
Comment:	The result is actually correct. The highlighted differences, as both execution results make sense from semantic perspective of the query. winner_name and T1.first_name , T1.last_name are equivalent for representation of “name". Evaluation failed because of the differences in the output format type. If multiple ground-truths are provided considering different output formats, this is not an error.
Human Expert	**True (Eval issue)**
Question 5:	**What are the different template type codes? (id=322, database=cre_Doc_Template_Mgt):**
Ground-truth:	`SELECT DISTINCT template_type_code FROM Templates`
SQL-PALM:	`SELECT DISTINCT template_type_code FROM Ref_Template_Types`
Comment:	The results are actually correct. *Few-shot SQL-PaLM* selects a different table Ref_Template_Types, instead of Templates. The same “template types” appear in multiple entries of Templates. The two SQL outputs generate the same execution result when evaluating with “DISTINCT"". Evaluation failed because Spider official evaluation removes ‘DISTINCT‘ during evaluation.
Human Expert	**True (Eval issue)**
Question 6:	**Find the number of professionals who have not treated any dogs. (id=983, database=dog_kennels):**
Ground-truth:	`SELECT count(*) FROM Professionals WHERE professional_id NOT IN ( SELECT professional_id FROM Treatments )`
SQL-PALM:	`SELECT count(*) FROM Professionals EXCEPT SELECT professional_id FROM Treatments`
Comment:	Left and Right sides of “’EXCEPT” need equivalent content. Corrected SQL is `SELECT count(*) FROM (SELECT professional_id FROM Professionals EXCEPT SELECT professional_id FROM Treatments)`
Human Expert	**False (Wrong Use of keywords)**
Question 7:	**Find the number of professionals who have not treated any dogs. (id=754, database=world_1):**
Ground-truth:	`select t1.name from country as t1 join countrylanguage as t2 on t1.code = t2.countrycode where t2.language = "english" and isofficial = "t" union select t1.name from country as t1 join countrylanguage as t2 on t1.code = t2.countrycode where t2.language = "dutch" and isofficial = "t"`
SQL-PALM:	`SELECT T1.name FROM country AS T1 JOIN countrylanguage AS T2 ON T1.code = T2.countrycode WHERE T2.language = "English" OR T2.language = "Dutch" AND T2.isofficial = "T"`
Comment:	Operator Precedence: ADD > OR. Need to add parenthesis over “OR”. Corrected SQL is `SELECT T1.name FROM country AS T1 JOIN countrylanguage AS T2 ON T1.code = T2.countrycode WHERE (T2.language = "English" OR T2.language = "Dutch") AND T2.isofficial = "T". Spider evaluation normalizes the ground-truth outputs to all lowercase for easier evaluation, but mismatch exists when referring to database content. Changes:english->English,dutch->Dutch, t->T`
Human Expert	**False (Wrong operator precedence and eval issue)**

Table 41: BIRD dev set errors from a sample of 100 queries denoted as "incorrect" by the evaluation procedure.

False-positive category	Number of examples
Wrong Ground-Truth	19
Wrong Evidence	5
Ambiguous	2
Evaluation Procedure	5

Table 42: These examples demonstrate the different categories of errors of *SQL-PaLM*-fine-tuned.

Wrong Ground-Truth	
Example 1:	
Question	**Write all comments made on the post titled ’How does gentle boosting differ from AdaBoost?’ (id=579, database=codebase_community)**
Ground-truth	`SELECT T1.Text FROM comments AS T1 INNER JOIN posts AS T2 ON T1.PostId = T2.Id WHERE T2.Title = ’How does gentle boosting differ FROM AdaBoost?’.`
Comment:	The ground-truth query has an upper case "FROM" instead of "from" which is what is in the question.
Example 2:	
Question	**What’s the finish time for the driver who ranked second in 2008’s Australian Grand Prix? (id=937, database=formula_1)**
Ground-truth	`SELECT T1.time FROM results AS T1 INNER JOIN races AS T2 on T1.raceId = T2.raceId WHERE T1.rank = 2 AND T2.name = ’Australian GrAND Prix’ AND T2.year = 2008`
Comment	Similarly to the previous example, the ground-truth query string doesn’t match the one from the question. Likely this is due to a data-cleaning procedure in the BirdSQL dev set.
Example 3:	
Question	**What race number has the most finishers? (id=979, database=formula_1)**
Ground-truth	`SELECT raceId FROM results GROUP BY raceId ORDER BY COUNT(time IS NOT NULL) DESC LIMIT 1`
Comment	The `COUNT(time IS NOT NULL)` is somewhat unconventional. Typically, `COUNT` is used on a column name directly. However, here it is counting the boolean result of `time IS NOT NULL`. This will count all rows, regardless of whether time is null or not, since the expression `time IS NOT NULL` is always either true or false, both of which are counted.
Example 4:	
Question	**Please provide top three football players’ IDs who are among the lowest potential players and prefer to use the right foot when attacking. (id=1135, database=european_football_2)**
Ground-truth	`SELECT id FROM Player_Attributes WHERE preferred_foot = ’right’ ORDER BY potential DESC LIMIT 3`
Comment	The questions asks the "lowest potential players" so the ground-query should order by descending `potential` - should not have `DESC`.
Wrong Evidence	
Example 1:	
Question	**What is the eligible free rate of the 10th and 11th schools with the highest enrolment for students in grades 1 through 12?’ (id=31, database=california_schools)**
Ground-truth	`SELECT CAST(‘Free Meal Count (K-12)‘ AS REAL) / ‘Enrollment (K-12)‘ FROM frpm ORDER BY ‘Enrollment (K-12)‘ DESC LIMIT 9, 2.`
Evidence	**K-12 refers to students in grades 1 through 12; Eligible free rate for K-12 = ‘FRPM Count (K-12)‘ / ‘Enrollment (K-12)‘**
Comment:	The evidence suggests that the information can be found in column `‘FRPM Count (K-12)‘` but we can see that in the ground-truth another column is actually chosen.
Example 2:	
Question	**Among all chemical compounds that contain molecule TR047, identify the percent that form a double-bond.’ (id=287, database=toxicology)**
Ground-truth	`SELECT CAST(COUNT(CASE WHEN T.bond_type = ’=’ THEN T.bond_id ELSE NULL END) AS REAL) * 100 / COUNT(T.bond_id) FROM bond AS T WHERE T.molecule_id = ’TR047’.`
Evidence	**TR047 is the molecule id; double bond refers to bond_type = ’ = ’; percentage = DIVIDE(SUM(bond_type = ’ = ’), COUNT(all bond_id)) as percent where molecule_id = ’TR047’‘**
Comment:	The evidence suggests that `bond_type` has spaces `’ = ’`, whereas the ground-truth query has no spaces.
Ambiguous	
Example 1:	
Question	**How many users last accessed the website after 2014/9/1? (id=533, database=codebase_community)**
Ground-truth	`SELECT COUNT(Id) FROM users WHERE date(LastAccessDate) > ’2014-09-01’.`
Comment:	It is not clear from the question whether we should include users that accessed the website exactly on the day of *2014/09/01*.
Evaluation Procedure	
Example 1:	
Question	**What is the height of the tallest player? Indicate his name. (id=1021, database=european_football_2)**
Ground-truth	`SELECT player_name FROM Player ORDER BY height DESC LIMIT 1`
SQL-PALM:	`SELECT height, player_name FROM Player ORDER BY height DESC LIMIT 1;`
Comment:	The generated query selects an additional column but is essentially equal to the ground-truth query.
Example 2:	
Question	**How many races were there in 2005? Name all the races in descending order. (id=592, database=formula_1**
Ground-truth	`SELECT name FROM races WHERE year = 2005 ORDER BY name DESC`
SQL-PALM:	`SELECT COUNT(raceId) FROM races WHERE year = 2005 UNION ALL SELECT name FROM races WHERE year = 2005 ORDER BY name DESC;`
Comment:	The generated query selects contains the correct selection of races and also includes the count with a `UNION ALL` statement.

A.10 Prompt examples

A.10.1 Concise Prompt Design: 4 shot

This is a task converting text into SQL statement. We will first given the dataset schema and then ask a question in text. You are asked to generate SQL statement.

Here is an example: Convert text to SQL:

[Schema (values)]: | farm | city : city_id , official_name , status , area_km_2 , population , census_ranking | farm : farm_id , year , total_horses , working_horses , total_cattle , oxen , bulls , cows , pigs , sheep_and_goats | farm_competition : competition_id , year , theme , host_city_id , hosts | competition_record : competition_id , farm_id , rank;

[Column names (type)]: city : city_id (number)| city : official_name (text)| city : status (text)| city : area_km_2 (number)| city : population (number)| city : census_ranking (text)| farm : farm_id (number)| farm : year (number)| farm : total_horses (number)| farm : working_horses (number)| farm : total_cattle (number)| farm : oxen (number)| farm : bulls (number)| farm : cows (number)| farm : pigs (number)| farm : sheep_and_goats (number)| farm_competition : competition_id (number)| farm_competition : year (number)| farm_competition : theme (text)| farm_competition : host_city_id (number)| farm_competition : hosts (text)| competition_record : competition_id (number)| competition_record : farm_id (number)| competition_record : rank (number);

[Primary Keys]: city : city_id | farm : farm_id | farm_competition : competition_id | competition_record : competition_id;

[Foreign Keys]: farm_competition : host_city_id equals city : city_id | competition_record : farm_id equals farm : farm_id | competition_record : competition_id equals farm_competition : competition_id

[Q]: What are the themes of farm competitions sorted by year in ascending order?;

[SQL]: select theme from farm_competition order by year asc;

Here is an example: Convert text to SQL:

[Schema (values)]: | farm | city : city_id , official_name , status , area_km_2 , population , census_ranking | farm : farm_id , year , total_horses , working_horses , total_cattle , oxen , bulls , cows , pigs , sheep_and_goats | farm_competition : competition_id , year , theme , host_city_id , hosts | competition_record : competition_id , farm_id , rank;

[Column names (type)]: city : city_id (number)| city : official_name (text)| city : status (text)| city : area_km_2 (number)| city : population (number)| city : census_ranking (text)| farm : farm_id (number)| farm : year (number)| farm : total_horses (number)| farm : working_horses (number)| farm : total_cattle (number)| farm : oxen (number)| farm : bulls (number)| farm : cows (number)| farm : pigs (number)| farm : sheep_and_goats (number)| farm_competition : competition_id (number)| farm_competition : year (number)| farm_competition : theme (text)| farm_competition : host_city_id (number)| farm_competition : hosts (text)| competition_record : competition_id (number)| competition_record : farm_id (number)| competition_record : rank (number);

[Primary Keys]: city : city_id | farm : farm_id | farm_competition : competition_id | competition_record : competition_id; [Foreign Keys]: farm_competition : host_city_id equals city : city_id | competition_record : farm_id equals farm : farm_id | competition_record : competition_id equals farm_competition : competition_id

[Q]: What are the maximum and minimum number of cows across all farms.;

[SQL]: select max(cows), min(cows) from farm;

Here is an example: Convert text to SQL:

[Schema (values)]: | department_management | department : department_id , name , creation , ranking , budget_in_billions , num_employees | head : head_id , name , born_state , age | management : department_id , head_id , temporary_acting (Yes);

[Column names (type)]: department : department_id (number)| department : name (text)| department : creation (text)| department : ranking (number)| department : budget_in_billions (number)| department : num_employees (number)| head : head_id (number)| head : name (text)| head : born_state (text)| head : age (number)| management : department_id (number)| management : head_id (number)| management : temporary_acting (text);

[Primary Keys]: department : department_id | head : head_id | management : department_id;

[Foreign Keys]: management : head_id equals head : head_id | management : department_id equals department : department_id

[Q]: Show the name and number of employees for the departments managed by heads whose temporary acting value is 'Yes'?;

[SQL]: select t1.name, t1.num_employees from department as t1 join management as t2 on t1.department_id = t2.department_id where t2.temporary_acting = 'Yes';

Here is an example: Convert text to SQL:

[Schema (values)]: | farm | city : city_id , official_name , status , area_km_2 , population , census_ranking | farm : farm_id , year , total_horses , working_horses , total_cattle , oxen , bulls , cows , pigs , sheep_and_goats | farm_competition : competition_id , year , theme , host_city_id , hosts | competition_record : competition_id , farm_id , rank;

[Column names (type)]: city : city_id (number)| city : official_name (text)| city : status (text)| city : area_km_2 (number)| city : population (number)| city : census_ranking (text)| farm : farm_id (number)| farm : year (number)| farm : total_horses (number)| farm : working_horses (number)| farm : total_cattle (number)| farm : oxen (number)| farm : bulls (number)| farm : cows (number)| farm : pigs (number)| farm : sheep_and_goats (number)| farm_competition : competition_id (number)| farm_competition : year (number)| farm_competition : theme (text)| farm_competition : host_city_id (number)| farm_competition : hosts (text)| competition_record : competition_id (number)| competition_record : farm_id (number)| competition_record : rank (number);

[Primary Keys]: city : city_id | farm : farm_id | farm_competition : competition_id | competition_record : competition_id;

[Foreign Keys]: farm_competition : host_city_id equals city : city_id | competition_record : farm_id equals farm : farm_id | competition_record : competition_id equals farm_competition : competition_id

[Q]: Show the status of the city that has hosted the greatest number of competitions.;

[SQL]: select t1.status from city as t1 join farm_competition as t2 on t1.city_id = t2.host_city_id group by t2.host_city_id order by count(*) desc limit 1;

Here is the test question to be answered: Convert text to SQL:

[Schema (values)]: | concert_singer | stadium : stadium_id , location , name , capacity , highest , lowest , average | singer : singer_id , name , country , song_name , song_release_year , age , is_male | concert : concert_id , concert_name , theme , stadium_id , year | singer_in_concert : concert_id , singer_id;

[Column names (type)]: stadium : stadium_id (number)| stadium : location (text)| stadium : name (text)| stadium : capacity (number)| stadium : highest (number)| stadium : lowest (number)|

stadium : average (number)| singer : singer_id (number)| singer : name (text)| singer : country (text)| singer : song_name (text)| singer : song_release_year (text)| singer : age (number)| singer : is_male (others)| concert : concert_id (number)| concert : concert_name (text)| concert : theme (text)| concert : stadium_id (text)| concert : year (text)| singer_in_concert : concert_id (number)| singer_in_concert : singer_id (text);

[Primary Keys]: stadium : stadium_id | singer : singer_id | concert : concert_id | singer_in_concert : concert_id;

[Foreign Keys]: concert : stadium_id equals stadium : stadium_id | singer_in_concert : singer_id equals singer : singer_id | singer_in_concert : concert_id equals concert : concert_id

[Q]: How many singers do we have?;

[SQL]:

A.10.2 Verbose Prompt Design: 4 shot

```
This is a task converting text into SQL statement. We will first given the
    dataset schema and then ask a question in text. You are asked to generate
    SQL statement.
Here is an example: Let us take a question and turn it into a SQL statement about
    database tables.  There are 4 tables
    . Their titles are: city, farm, farm_competition, competition_record. Table
     1 is city, and its column names and types are: City_ID (Type is number),
    Official_Name (Type is text), Status (Type is text), Area_km_2 (Type is
    number), Population (Type is number), Census_Ranking (Type is text). Table
    2 is farm, and its column names and types are: Farm_ID (Type is number),
    Year (Type is number), Total_Horses (Type is number), Working_Horses (Type
    is number), Total_Cattle (Type is number), Oxen (Type is number), Bulls (
    Type is number), Cows (Type is number), Pigs (Type is number),
    Sheep_and_Goats (Type is number). Table 3 is farm_competition, and its
    column names and types are: Competition_ID (Type is number), Year (Type is
    number), Theme (Type is text), Host_city_ID (Type is number), Hosts (Type
    is text). Table 4 is competition_record, and its column names and types are
    : Competition_ID (Type is number), Farm_ID (Type is number), Rank (Type is
    number).
    The primary keys are: city_id from Table city, farm_id from Table farm,
    competition_id from Table farm_competition, competition_id from Table
    competition_record.
    The foreign keys are: host_city_id from Table farm_competition is equivalent
    with city_id from Table city, farm_id from Table competition_record is
    equivalent with farm_id from Table farm, competition_id from Table
    competition_record is equivalent with competition_id from Table
    farm_competition. Use foreign keys to join Tables.  Let us take a text
    question and turn it into a SQL statement about database tables. The
    question is: What are the themes of farm competitions sorted by year in
    ascending order? The corresponding SQL is: SELECT Theme FROM
    farm_competition ORDER BY YEAR ASC;
Here is an example: Let us take a question and turn it into a SQL statement about
    database tables.  There are 4 tables
    . Their titles are: city, farm, farm_competition, competition_record. Table
     1 is city, and its column names and types are: City_ID (Type is number),
    Official_Name (Type is text), Status (Type is text), Area_km_2 (Type is
    number), Population (Type is number), Census_Ranking (Type is text). Table
    2 is farm, and its column names and types are: Farm_ID (Type is number),
    Year (Type is number), Total_Horses (Type is number), Working_Horses (Type
    is number), Total_Cattle (Type is number), Oxen (Type is number), Bulls (
    Type is number), Cows (Type is number), Pigs (Type is number),
    Sheep_and_Goats (Type is number). Table 3 is farm_competition, and its
    column names and types are: Competition_ID (Type is number), Year (Type is
    number), Theme (Type is text), Host_city_ID (Type is number), Hosts (Type
    is text). Table 4 is competition_record, and its column names and types are
    : Competition_ID (Type is number), Farm_ID (Type is number), Rank (Type is
    number).
    The primary keys are: city_id from Table city, farm_id from Table farm,
    competition_id from Table farm_competition, competition_id from Table
    competition_record.
    The foreign keys are: host_city_id from Table farm_competition is equivalent
    with city_id from Table city, farm_id from Table competition_record is
    equivalent with farm_id from Table farm, competition_id from Table
    competition_record is equivalent with competition_id from Table
    farm_competition. Use foreign keys to join Tables.  Let us take a text
    question and turn it into a SQL statement about database tables. The
    question is: What are the maximum and minimum number of cows across all
    farms. The corresponding SQL is: SELECT max(Cows) ,  min(Cows) FROM farm;
Here is an example: Let us take a question and turn it into a SQL statement about
    database tables.  There are 3 tables
    . Their titles are: department, head, management. Table 1 is department,
    and its column names and types are: Department_ID (Type is number), Name (
    Type is text), Creation (Type is text), Ranking (Type is number),
    Budget_in_Billions (Type is number), Num_Employees (Type is number). Table
    2 is head, and its column names and types are: head_ID (Type is number),
    name (Type is text), born_state (Type is text), age (Type is number). Table
     3 is management, and its column names and types are: department_ID (Type
    is number), head_ID (Type is number), temporary_acting (Type is text).
    The primary keys are: department_id from Table department, head_id from Table
    head, department_id from Table management.
    The foreign keys are: head_id from Table management is equivalent with head_id
    from Table head, department_id from Table management is equivalent with
    department_id from Table department. Use foreign keys to join Tables.
    Columns with relevant values: Table management Column temporary_acting have
     values: Yes;  Only use columns with relevant values to generate SQL.  Let
    us take a text question and turn it into a SQL statement about database
    tables. The question is: Show the name and number of employees for the
    departments managed by heads whose temporary acting value is 'Yes'? The
    corresponding SQL is: SELECT T1.name ,  T1.num_employees FROM department AS
     T1 JOIN management AS T2 ON T1.department_id  =  T2.department_id WHERE T2
    .temporary_acting  =  'Yes';
Here is an example: Let us take a question and turn it into a SQL statement about
    database tables.  There are 4 tables
    . Their titles are: city, farm, farm_competition, competition_record. Table
     1 is city, and its column names and types are: City_ID (Type is number),
    Official_Name (Type is text), Status (Type is text), Area_km_2 (Type is
    number), Population (Type is number), Census_Ranking (Type is text). Table
    2 is farm, and its column names and types are: Farm_ID (Type is number),
    Year (Type is number), Total_Horses (Type is number), Working_Horses (Type
    is number), Total_Cattle (Type is number), Oxen (Type is number), Bulls (
    Type is number), Cows (Type is number), Pigs (Type is number),
    Sheep_and_Goats (Type is number). Table 3 is farm_competition, and its
    column names and types are: Competition_ID (Type is number), Year (Type is
    number), Theme (Type is text), Host_city_ID (Type is number), Hosts (Type
    is text). Table 4 is competition_record, and its column names and types are
    : Competition_ID (Type is number), Farm_ID (Type is number), Rank (Type is
    number).
    The primary keys are: city_id from Table city, farm_id from Table farm,
    competition_id from Table farm_competition, competition_id from Table
    competition_record.
    The foreign keys are: host_city_id from Table farm_competition is equivalent
    with city_id from Table city, farm_id from Table competition_record is
    equivalent with farm_id from Table farm, competition_id from Table
    competition_record is equivalent with competition_id from Table
    farm_competition. Use foreign keys to join Tables.  Let us take a text
    question and turn it into a SQL statement about database tables. The
    question is: Show the status of the city that has hosted the greatest
    number of competitions. The corresponding SQL is: SELECT T1.Status FROM
    city AS T1 JOIN farm_competition AS T2 ON T1.City_ID  =  T2.Host_city_ID
    GROUP BY T2.Host_city_ID ORDER BY COUNT(*) DESC LIMIT 1;
 Here is the test question to be answered: Let us take a question and turn it into a
     SQL statement about database tables.
     There are 4 tables. Their titles are: stadium, singer, concert,
     singer_in_concert. Table 1 is stadium, and its column names and types are:
      Stadium_ID (Type is number), Location (Type is text), Name (Type is text)
     , Capacity (Type is number), Highest (Type is number), Lowest (Type is
     number), Average (Type is number). Table 2 is singer, and its column names
      and types are: Singer_ID (Type is number), Name (Type is text), Country (
     Type is text), Song_Name (Type is text), Song_release_year (Type is text),
      Age (Type is number), Is_male (Type is others). Table 3 is concert, and
     its column names and types are: concert_ID (Type is number), concert_Name
     (Type is text), Theme (Type is text), Stadium_ID (Type is text), Year (
     Type is text). Table 4 is singer_in_concert, and its column names and
     types are: concert_ID (Type is number), Singer_ID (Type is text).
     The primary keys are: stadium_id from Table stadium, singer_id from Table
     singer, concert_id from Table concert, concert_id from Table
     singer_in_concert.
     The foreign keys are: stadium_id from Table concert is equivalent with
     stadium_id from Table stadium, singer_id from Table singer_in_concert is
     equivalent with singer_id from Table singer, concert_id from Table
     singer_in_concert is equivalent with concert_id from Table concert. Use
     foreign keys to join Tables.  Let us take a text question and turn it into
      a SQL statement about database tables. The question is: How many singers
     do we have? The corresponding SQL is:
```

A.11 Database content

See “[Database values that related with questions]:” in red to show database content values.

```
Here is the test question to be anwered: Convert text to SQL:
 [Schema (values)]: | california_schools | frpm : CDSCode , Academic Year ,
     County Code , District Code , School Code , County Name , District Name ,
     School Name , District Type , School Type , Educational Option Type , NSLP
     Provision Status , Charter School (Y/N) , Charter School Number , Charter
     Funding Type , IRC , Low Grade , High Grade , Enrollment (K-12) , Free Meal
      Count (K-12) , Percent (%) Eligible Free (K-12) , FRPM Count (K-12) ,
     Percent (%) Eligible FRPM (K-12) , Enrollment (Ages 5-17) , Free Meal Count
      (Ages 5-17) , Percent (%) Eligible Free (Ages 5-17) , FRPM Count (Ages
     5-17) , Percent (%) Eligible FRPM (Ages 5-17) , 2013-14 CALPADS Fall 1
     Certification Status | satscores : cds , rtype , sname , dname , cname ,
     enroll12 , NumTstTakr , AvgScrRead , AvgScrMath , AvgScrWrite , NumGE1500 |
      schools : CDSCode , NCESDist , NCESSchool , StatusType , County , District
      , School , Street , StreetAbr , City , Zip , State , MailStreet ,
     MailStrAbr , MailCity , MailZip , MailState , Phone , Ext , Website ,
     OpenDate , ClosedDate , Charter , CharterNum , FundingType , DOC , DOCType
     , SOC , SOCType , EdOpsCode , EdOpsName , EILCode , EILName , GSoffered ,
     GSserved , Virtual , Magnet , Latitude , Longitude , AdmFName1 , AdmLName1
     , AdmEmail1 , AdmFName2 , AdmLName2 , AdmEmail2 , AdmFName3 , AdmLName3 ,
     AdmEmail3 , LastUpdate;
  [Column names (type)] : frpm : cdscode (text) | frpm : academic year (text) |
     frpm : county code (text) | frpm : district code (number) | frpm : school
     code (text) | frpm : county name (text) | frpm : district name (text) |
     frpm : school name (text) | frpm : district type (text) | frpm : school
     type (text) | frpm : educational option type (text) | frpm : nslp provision
      status (text) | frpm : charter school (y/n) (number) | frpm : charter
     school number (text) | frpm : charter funding type (text) | frpm : irc (
     number) | frpm : low grade (text) | frpm : high grade (text) | frpm :
     enrollment (k-12) (number) | frpm : free meal count (k-12) (number) | frpm
     : percent (%) eligible free (k-12) (number) | frpm : frpm count (k-12) (
     number) | frpm : percent (%) eligible frpm (k-12) (number) | frpm :
     enrollment (ages 5-17) (number) | frpm : free meal count (ages 5-17) (
     number) | frpm : percent (%) eligible free (ages 5-17) (number) | frpm :
     frpm count (ages 5-17) (number) | frpm : percent (%) eligible frpm (ages
     5-17) (number) | frpm : 2013-14 calpads fall 1 certification status (number
     ) | satscores : cds (text) | satscores : rtype (text) | satscores : sname (
     text) | satscores : dname (text) | satscores : cname (text) | satscores :
     enroll12 (number) | satscores : numtsttakr (number) | satscores :
     avgscrread (number) | satscores : avgscrmath (number) | satscores :
     avgscrwrite (number) | satscores : numge1500 (number) | schools : cdscode (
     text) | schools : ncesdist (text) | schools : ncesschool (text) | schools :
      statustype (text) | schools : county (text) | schools : district (text) |
     schools : school (text) | schools : street (text) | schools : streetabr (
     text) | schools : city (text) | schools : zip (text) | schools : state (
     text) | schools : mailstreet (text) | schools : mailstrabr (text) | schools
      : mailcity (text) | schools : mailzip (text) | schools : mailstate (text)
     | schools : phone (text) | schools : ext (text) | schools : website (text)
     | schools : opendate (time) | schools : closeddate (time) | schools :
     charter (number) | schools : charternum (text) | schools : fundingtype (
     text) | schools : doc (text) | schools : doctype (text) | schools : soc (
     text) | schools : soctype (text) | schools : edopscode (text) | schools :
     edopsname (text) | schools : eilcode (text) | schools : eilname (text) |
     schools : gsoffered (text) | schools : gsserved (text) | schools : virtual
     (text) | schools : magnet (number) | schools : latitude (number) | schools
     : longitude (number) | schools : admfname1 (text) | schools : admlname1 (
     text) | schools : admemail1 (text) | schools : admfname2 (text) | schools :
      admlname2 (text) | schools : admemail2 (text) | schools : admfname3 (text)
      | schools : admlname3 (text) | schools : admemail3 (text) | schools :
     lastupdate (time);
  [Primary Keys]: frpm : CDSCode | satscores : cds | schools : CDSCode;
  [Foreign Keys] : frpm : CDSCode equals schools : CDSCode | satscores : cds
     equals schools : CDSCode;
  [Database values that related with questions]:
  The column ‘County Name‘ in Table ‘frpm‘ has database values:  Alameda
The column ‘cname‘ in Table ‘satscores‘ has database values:  Alameda
The column ‘County‘ in Table ‘schools‘ has database values:  Alameda
The column ‘City‘ in Table ‘schools‘ has database values:  Alameda
The column ‘MailCity‘ in Table ‘schools‘ has database values:  Alameda
The column ‘GSoffered‘ in Table ‘schools‘ has database values:  K-12
The column ‘GSserved‘ in Table ‘schools‘ has database values:  K-12
The column ‘AdmFName1‘ in Table ‘schools‘ has database values:  Rae
The column ‘AdmLName1‘ in Table ‘schools‘ has database values:  Free
;
```

```
 [Additional Info]: Eligible free rate for K-12 = `FRPM Count (K-12)` / `
    Enrollment (K-12)`
  [Q]: What is the highest eligible free rate for K-12 students in the schools
     in Alameda County?;
  [SQL]:
 Here is an example: Convert text to SQL:
```

A.11.1 Full column description

See "[detailed description of tables and columns]" in red to show entire column descriptions, and they are very lengthy.

```
 Here is the test question to be anwered: Convert text to SQL:
  [Schema (values)]: | california_schools | frpm : CDSCode , Academic Year ,
     County Code , District Code , School Code , County Name , District Name ,
     School Name , District Type , School Type , Educational Option Type , NSLP
      Provision Status , Charter School (Y/N) , Charter School Number , Charter
      Funding Type , IRC , Low Grade , High Grade , Enrollment (K-12) , Free
     Meal Count (K-12) , Percent (%) Eligible Free (K-12) , FRPM Count (K-12) ,
      Percent (%) Eligible FRPM (K-12) , Enrollment (Ages 5-17) , Free Meal
     Count (Ages 5-17) , Percent (%) Eligible Free (Ages 5-17) , FRPM Count (
     Ages 5-17) , Percent (%) Eligible FRPM (Ages 5-17) , 2013-14 CALPADS Fall
     1 Certification Status | satscores : cds , rtype , sname , dname , cname ,
      enroll12 , NumTstTakr , AvgScrRead , AvgScrMath , AvgScrWrite , NumGE1500
      | schools : CDSCode , NCESDist , NCESSchool , StatusType , County ,
     District , School , Street , StreetAbr , City , Zip , State , MailStreet ,
      MailStrAbr , MailCity , MailZip , MailState , Phone , Ext , Website ,
     OpenDate , ClosedDate , Charter , CharterNum , FundingType , DOC , DOCType
      , SOC , SOCType , EdOpsCode , EdOpsName , EILCode , EILName , GSoffered ,
      GSserved , Virtual , Magnet , Latitude , Longitude , AdmFName1 ,
     AdmLName1 , AdmEmail1 , AdmFName2 , AdmLName2 , AdmEmail2 , AdmFName3 ,
     AdmLName3 , AdmEmail3 , LastUpdate;
  [Column names (type)]: frpm : CDSCode (text) | frpm : Academic Year (text) |
     frpm : County Code (text) | frpm : District Code (number) | frpm : School
     Code (text) | frpm : County Name (text) | frpm : District Name (text) |
     frpm : School Name (text) | frpm : District Type (text) | frpm : School
     Type (text) | frpm : Educational Option Type (text) | frpm : NSLP
     Provision Status (text) | frpm : Charter School (Y/N) (number) | frpm :
     Charter School Number (text) | frpm : Charter Funding Type (text) | frpm :
      IRC (number) | frpm : Low Grade (text) | frpm : High Grade (text) | frpm
     : Enrollment (K-12) (number) | frpm : Free Meal Count (K-12) (number) |
     frpm : Percent (%) Eligible Free (K-12) (number) | frpm : FRPM Count (K
     -12) (number) | frpm : Percent (%) Eligible FRPM (K-12) (number) | frpm :
     Enrollment (Ages 5-17) (number) | frpm : Free Meal Count (Ages 5-17) (
     number) | frpm : Percent (%) Eligible Free (Ages 5-17) (number) | frpm :
     FRPM Count (Ages 5-17) (number) | frpm : Percent (%) Eligible FRPM (Ages
     5-17) (number) | frpm : 2013-14 CALPADS Fall 1 Certification Status (
     number) | satscores : cds (text) | satscores : rtype (text) | satscores :
     sname (text) | satscores : dname (text) | satscores : cname (text) |
     satscores : enroll12 (number) | satscores : NumTstTakr (number) |
     satscores : AvgScrRead (number) | satscores : AvgScrMath (number) |
     satscores : AvgScrWrite (number) | satscores : NumGE1500 (number) |
     schools : CDSCode (text) | schools : NCESDist (text) | schools :
     NCESSchool (text) | schools : StatusType (text) | schools : County (text)
     | schools : District (text) | schools : School (text) | schools : Street (
     text) | schools : StreetAbr (text) | schools : City (text) | schools : Zip
      (text) | schools : State (text) | schools : MailStreet (text) | schools :
```

```
     MailStrAbr (text) | schools : MailCity (text) | schools : MailZip (text)
    | schools : MailState (text) | schools : Phone (text) | schools : Ext (
    text) | schools : Website (text) | schools : OpenDate (time) | schools :
    ClosedDate (time) | schools : Charter (number) | schools : CharterNum (
    text) | schools : FundingType (text) | schools : DOC (text) | schools :
    DOCType (text) | schools : SOC (text) | schools : SOCType (text) | schools
     : EdOpsCode (text) | schools : EdOpsName (text) | schools : EILCode (text
    ) | schools : EILName (text) | schools : GSoffered (text) | schools :
    GSserved (text) | schools : Virtual (text) | schools : Magnet (number) |
    schools : Latitude (number) | schools : Longitude (number) | schools :
    AdmFName1 (text) | schools : AdmLName1 (text) | schools : AdmEmail1 (text)
     | schools : AdmFName2 (text) | schools : AdmLName2 (text) | schools :
    AdmEmail2 (text) | schools : AdmFName3 (text) | schools : AdmLName3 (text)
     | schools : AdmEmail3 (text) | schools : LastUpdate (time);
[Primary Keys]: frpm : CDSCode | satscores : cds | schools : CDSCode;
 [Foreign Keys]: frpm : CDSCode equals schools : CDSCode | satscores : cds equals
    schools : CDSCode;
 [detailed description of tables and columns]:
Column description of Table "frpm" have the following descriptions:
 Column "County Name" of Table "frpm", means "County Code"
Column "Charter School (Y/N)" of Table frpm has value descriptions "0: N;1: Y"
Column "IRC" of Table frpm has value descriptions "Not useful"
Column "Enrollment (K-12)" of Table frpm has value descriptions "commonsense
   evidence:K-12: 1st grade - 12nd grade"
Column "Free Meal Count (K-12)" of Table frpm has value descriptions "
   commonsense evidence:eligible free rate = Free Meal Count / Enrollment"
Column "FRPM Count (K-12)" of Table "frpm", means "Free or Reduced Price Meal
   Count (K-12)", has value descriptions "commonsense evidence:eligible FRPM
   rate = FRPM / Enrollment"
Column "Free Meal Count (Ages 5-17)" of Table frpm has value descriptions "
   commonsense evidence:eligible free rate = Free Meal Count / Enrollment"
Column description of Table "satscores" have the following descriptions:
Column "cds" of Table "satscores", means "California Department Schools"
Column "rtype" of Table satscores has value descriptions "unuseful"
Column "sname" of Table "satscores", means "school name"
Column "dname" of Table "satscores", means "district segment", district name,
Column "cname" of Table "satscores", means "county name"
Column "enroll12" of Table "satscores", means "enrollment (1st-12nd grade)"
Column "NumTstTakr" of Table "satscores", means "Number of Test Takers in this
   school", Number of Test Takers, , has value descriptions "number of test
   takers in each school"
Column "AvgScrRead" of Table "satscores", means "average scores in Reading"
Column "AvgScrMath" of Table "satscores", means "average scores in Math"
Column "AvgScrWrite" of Table "satscores", means "average scores in writing"
Column "NumGE1500" of Table "satscores", means "Number of Test Takers Whose
   Total SAT Scores Are Greater or Equal to 1500", has value descriptions "
   Number of Test Takers Whose Total SAT Scores Are Greater or Equal to 1500
   commonsense evidence:Excellence Rate = NumGE1500 / NumTstTakr"
Column description of Table "schools" have the following descriptions:
Column "NCESDist" of Table "schools", means "This field represents the 7-digit
   National Center for Educational Statistics (NCES) school district
   identification number. The first 2 digits identify the state and the last 5
    digits identify the school district. Combined, they make a unique 7-digit
   ID for each school district.", National Center for Educational Statistics
   school district identification number,
Column "NCESSchool" of Table "schools", means "This field represents the 5-
   digit NCES school identification number. The NCESSchool combined with the
```

```
    NCESDist form a unique 12-digit ID for each school.", National Center for
    Educational Statistics school identification number,
Column "StatusType" of Table "schools", means "This field identifies the status
     of the district.", has value descriptions "Definitions of the valid status
     types are listed below:      Active: The district is in operation and
    providing instructional services.      Closed: The district is not in
    operation and no longer providing instructional services.      Merged: The
     district has combined with another district or districts.      Pending:
    The district has not opened for operation and instructional services yet,
    but plans to open within the next 912 months."
Column "County" of Table "schools", means "County name"
Column "StreetAbr" of Table "schools", means "The abbreviated street address of
     the school, district, or administrative authoritys physical location.",
    street address, , has value descriptions "The abbreviated street address of
     the school, district, or administrative authoritys physical location. Note
    : Some records (primarily records of closed or retired schools) may not
    have data in this field."
Column "MailStreet" of Table schools has value descriptions "The unabbreviated
    mailing address of the school, district, or administrative authority. Note:
     1) Some entities (primarily closed or retired schools) may not have data
    in this field; 2) Many active entities have not provided a mailing street
    address. For your convenience we have filled the unpopulated MailStreet
    cells with Street data."
Column "MailStrAbr" of Table "schools", means "mailing street address", has
    value descriptions "the abbreviated mailing street address of the school,
    district, or administrative authority.Note: Many active entities have not
    provided a mailing street address. For your convenience we have filled the
    unpopulated MailStrAbr cells with StreetAbr data."
Column "MailCity" of Table "schools", means "mailing city", has value
    descriptions "The city associated with the mailing address of the school,
    district, or administrative authority. Note: Many entities have not
    provided a mailing address city. For your convenience we have filled the
    unpopulated MailCity cells with City data."
Column "MailZip" of Table "schools", means "mailing zip", has value
    descriptions "The zip code associated with the mailing address of the
    school, district, or administrative authority. Note: Many entities have not
     provided a mailing address zip code. For your convenience we have filled
    the unpopulated MailZip cells with Zip data."
Column "MailState" of Table "schools", means "mailing state", has value
    descriptions "The state within the mailing address. For your convenience we
     have filled the unpopulated MailState cells with State data."
Column "Ext" of Table "schools", means "The phone number extension of the
    school, district, or administrative authority.", extension,
Column "Website" of Table "schools", means "The website address of the school,
    district, or administrative authority."
Column "OpenDate" of Table "schools", means "The date the school opened."
Column "ClosedDate" of Table "schools", means "The date the school closed."
Column "Charter" of Table "schools", means "This field identifies a charter
    school.", has value descriptions "The field is coded as follows: 1 = The
    school is a charter 0 = The school is not a charter"
Column "CharterNum" of Table "schools", means "The charter school number,", has
     value descriptions "4-digit number assigned to a charter school."
Column "FundingType" of Table "schools", means "Indicates the charter school
    funding type", has value descriptions "Values are as follows: Not in CS (
    California School) funding model Locally funded Directly funded"
Column "DOC" of Table "schools", means "District Ownership Code", has value
    descriptions "The District Ownership Code (DOC) is the numeric code used to
     identify the category of the Administrative Authority.      00 - County
    Office of Education       02  State Board of Education       03  Statewide
    Benefit Charter       31  State Special Schools       34  Non-school
    Location       52  Elementary School District       54  Unified School
    District       56  High School District       98  Regional Occupational
    Center/Program (ROC/P)commonsense evidence:Only the California Education
    Authority has been included in the non-school location category."
Column "DOCType" of Table "schools", means "The District Ownership Code Type is
     the text description of the DOC category.", The District Ownership Code
    Type, , has value descriptions "(See text values in DOC field description
    above)"
Column "SOC" of Table "schools", means "The School Ownership Code is a numeric
    code used to identify the type of school.", School Ownership Code, , has
    value descriptions "08 - Preschool             09  Special Education
    Schools (Public)      11  Youth Authority Facilities (CEA)       13
    Opportunity Schools       14  Juvenile Court Schools       15  Other County
     or District Programs       31  State Special Schools       60  Elementary
    School (Public)       61  Elementary School in 1 School District (Public)
          62  Intermediate/Middle Schools (Public)       63  Alternative
    Schools of Choice       64  Junior High Schools (Public)       65  K-12
    Schools (Public)       66  High Schools (Public)       67  High Schools in
    1 School District (Public)       68  Continuation High Schools       69
    District Community Day Schools       70  Adult Education Centers       98
    Regional Occupational Center/Program (ROC/P)"
Column "SOCType" of Table "schools", means "The School Ownership Code Type is
    the text description of the type of school.", School Ownership Code Type,
Column "EdOpsCode" of Table "schools", means "The Education Option Code is a
    short text description of the type of education offered.", Education Option
     Code, , has value descriptions "ALTSOC  Alternative School of Choice
    COMM  County Community School       COMMDAY  Community Day School       CON
      Continuation School       JUV  Juvenile Court School       OPP
    Opportunity School       YTH  Youth Authority School       SSS  State
    Special School       SPEC  Special Education School       TRAD  Traditional
           ROP  Regional Occupational Program       HOMHOS  Home and Hospital
          SPECON  District Consortia Special Education School"
Column "EdOpsName" of Table "schools", means "Educational Option Name", has
    value descriptions "The Educational Option Name is the long text
    description of the type of education being offered."
Column "EILCode" of Table "schools", means "The Educational Instruction Level
    Code is a short text description of the institution's type relative to the
    grade range served.", Educational Instruction Level Code, , has value
    descriptions "A  Adult       ELEM  Elementary       ELEMHIGH  Elementary-
    High Combination       HS  High School       INTMIDJR  Intermediate/Middle/
    Junior High       PS  Preschool       UG  Ungraded"
Column "EILName" of Table "schools", means "The Educational Instruction Level
    Name is the long text description of the institutions type relative to the
    grade range served.", Educational Instruction Level Name,
Column "GSoffered" of Table "schools", means "The grade span offered is the
    lowest grade and the highest grade offered or supported by the school,
    district, or administrative authority. This field might differ from the
    grade span served as reported in the most recent certified California
    Longitudinal Pupil Achievement (CALPADS) Fall 1 data collection.", grade
    span offered, , has value descriptions "For example XYZ School might
    display the following data:GSoffered = PAdultGSserved = K12"
Column "GSserved" of Table "schools", means "It is the lowest grade and the
    highest grade of student enrollment as reported in the most recent
    certified CALPADS Fall 1 data collection. Only K12 enrollment is reported
    through CALPADS. This field may differ from the grade span offered.", grade
     span served., , has value descriptions "commonsense evidence:1. Only K12
    enrollment is reported through CALPADS2. Note: Special programs at
    independent study, alternative education, and special education schools
    will often exceed the typical grade span for schools of that type"
Column "Virtual" of Table "schools", means "This field identifies the type of
    virtual instruction offered by the school. Virtual instruction is
    instruction in which students and teachers are separated by time and/or
    location, and interaction occurs via computers and/or telecommunications
    technologies.", has value descriptions "The field is coded as follows: F =
    Exclusively Virtual  The school has no physical building where students
    meet with each other or with teachers, all instruction is virtual. V =
    Primarily Virtual  The school focuses on a systematic program of virtual
    instruction but includes some physical meetings among students or with
    teachers. C = Primarily Classroom  The school offers virtual courses but
    virtual instruction is not the primary means of instruction. N = Not
    Virtual  The school does not offer any virtual instruction. P = Partial
    Virtual  The school offers some, but not all, instruction through virtual
    instruction. Note: This value was retired and replaced with the Primarily
    Virtual and Primarily Classroom values beginning with the 201617 school
    year."
Column "Magnet" of Table "schools", means "This field identifies whether a
    school is a magnet school and/or provides a magnet program.", has value
    descriptions "The field is coded as follows: Y = Magnet - The school is a
    magnet school and/or offers a magnet program. N = Not Magnet - The school
    is not a magnet school and/or does not offer a magnet program.commonsense
    evidence:Note: Preschools and adult education centers do not contain a
    magnet school indicator."
Column "Latitude" of Table "schools", means "The angular distance (expressed in
     degrees) between the location of the school, district, or administrative
    authority and the equator measured north to south."
Column "Longitude" of Table "schools", means "The angular distance (expressed
    in degrees) between the location of the school, district, or administrative
     authority and the prime meridian (Greenwich, England) measured from west
    to east."
Column "AdmFName1" of Table "schools", means "administrator's first name", has
    value descriptions "The superintendents or principals first name.
    commonsense evidence:Only active and pending districts and schools will
    display administrator information, if applicable."
Column "AdmLName1" of Table "schools", means "administrator's last name", has
    value descriptions "The superintendents or principals last name.commonsense
     evidence:Only active and pending districts and schools will display
    administrator information, if applicable."
Column "AdmEmail1" of Table "schools", means "administrator's email address",
    has value descriptions "The superintendents or principals email address.
    commonsense evidence:Only active and pending districts and schools will
    display administrator information, if applicable."
Column "AdmFName2" of Table schools has value descriptions "SAME as 1"
Column "AdmFName3" of Table schools has value descriptions "not useful"
Column "AdmLName3" of Table schools has value descriptions "not useful"
Column "AdmEmail3" of Table schools has value descriptions "not useful"
Column "LastUpdate" of Table schools has value descriptions "when is this
    record updated last time"
;
 [Database values that related with questions]
The column `County Name` in Table `frpm` has database values: Alameda
The column `cname` in Table `satscores` has database values: Alameda
The column `County` in Table `schools` has database values: Alameda
The column `City` in Table `schools` has database values: Alameda
The column `MailCity` in Table `schools` has database values: Alameda
The column `GSoffered` in Table `schools` has database values: K-12
The column `GSserved` in Table `schools` has database values: K-12
The column `AdmFName1` in Table `schools` has database values: Rae
The column `AdmLName1` in Table `schools` has database values: Free
;
 [Additional Info]: Eligible free rate for K-12 = `FRPM Count (K-12)` / `
    Enrollment (K-12)`
 [Q]: What is the highest eligible free rate for K-12 students in the schools
    in Alameda County?;
 [SQL]:
Here is an example: Convert text to SQL:
```

A.11.2 Inferred column selection

See "[detailed description of tables and columns]" in red for soft column selection.

```
Here is the test question to be anwered: Convert text to SQL:
 [Schema (values)]: | california_schools | frpm : CDSCode , Academic Year ,
   County Code , District Code , School Code , County Name , District Name ,
   School Name , District Type , School Type , Educational Option Type , NSLP
   Provision Status , Charter School (Y/N) , Charter School Number , Charter
   Funding Type , IRC , Low Grade , High Grade , Enrollment (K-12) , Free Meal
    Count (K-12) , Percent (%) Eligible Free (K-12) , FRPM Count (K-12) ,
   Percent (%) Eligible FRPM (K-12) , Enrollment (Ages 5-17) , Free Meal Count
    (Ages 5-17) , Percent (%) Eligible Free (Ages 5-17) , FRPM Count (Ages
   5-17) , Percent (%) Eligible FRPM (Ages 5-17) , 2013-14 CALPADS Fall 1
   Certification Status | satscores : cds , rtype , sname , dname , cname ,
   enroll12 , NumTstTakr , AvgScrRead , AvgScrMath , AvgScrWrite , NumGE1500 |
    schools : CDSCode , NCESDist , NCESSchool , StatusType , County , District
    , School , Street , StreetAbr , City , Zip , State , MailStreet ,
   MailStrAbr , MailCity , MailZip , MailState , Phone , Ext , Website ,
   OpenDate , ClosedDate , Charter , CharterNum , FundingType , DOC , DOCType
   , SOC , SOCType , EdOpsCode , EdOpsName , EILCode , EILName , GSoffered ,
   GSserved , Virtual , Magnet , Latitude , Longitude , AdmFName1 , AdmLName1
   , AdmEmail1 , AdmFName2 , AdmLName2 , AdmEmail2 , AdmFName3 , AdmLName3 ,
   AdmEmail3 , LastUpdate;
 [Column names (type)]: frpm : CDSCode (text) | frpm : Academic Year (text) | frpm
    : County Code (text) | frpm : District Code (number) | frpm : School Code
   (text) | frpm : County Name (text) | frpm : District Name (text) | frpm :
   School Name (text) | frpm : District Type (text) | frpm : School Type (text
   ) | frpm : Educational Option Type (text) | frpm : NSLP Provision Status (
   text) | frpm : Charter School (Y/N) (number) | frpm : Charter School Number
    (text) | frpm : Charter Funding Type (text) | frpm : IRC (number) | frpm :
    Low Grade (text) | frpm : High Grade (text) | frpm : Enrollment (K-12) (
   number) | frpm : Free Meal Count (K-12) (number) | frpm : Percent (%)
   Eligible Free (K-12) (number) | frpm : FRPM Count (K-12) (number) | frpm :
   Percent (%) Eligible FRPM (K-12) (number) | frpm : Enrollment (Ages 5-17) (
   number) | frpm : Free Meal Count (Ages 5-17) (number) | frpm : Percent (%)
   Eligible Free (Ages 5-17) (number) | frpm : FRPM Count (Ages 5-17) (number)
    | frpm : Percent (%) Eligible FRPM (Ages 5-17) (number) | frpm : 2013-14
   CALPADS Fall 1 Certification Status (number) | satscores : cds (text) |
   satscores : rtype (text) | satscores : sname (text) | satscores : dname (
   text) | satscores : cname (text) | satscores : enroll12 (number) |
   satscores : NumTstTakr (number) | satscores : AvgScrRead (number) |
   satscores : AvgScrMath (number) | satscores : AvgScrWrite (number) |
   satscores : NumGE1500 (number) | schools : CDSCode (text) | schools :
    NCESDist (text) | schools : NCESSchool (text) | schools : StatusType (text)
     | schools : County (text) | schools : District (text) | schools : School (
    text) | schools : Street (text) | schools : StreetAbr (text) | schools :
    City (text) | schools : Zip (text) | schools : State (text) | schools :
    MailStreet (text) | schools : MailStrAbr (text) | schools : MailCity (text)
     | schools : MailZip (text) | schools : MailState (text) | schools : Phone
    (text) | schools : Ext (text) | schools : Website (text) | schools :
    OpenDate (time) | schools : ClosedDate (time) | schools : Charter (number)
    | schools : CharterNum (text) | schools : FundingType (text) | schools :
    DOC (text) | schools : DOCType (text) | schools : SOC (text) | schools :
    SOCType (text) | schools : EdOpsCode (text) | schools : EdOpsName (text) |
    schools : EILCode (text) | schools : EILName (text) | schools : GSoffered (
    text) | schools : GSserved (text) | schools : Virtual (text) | schools :
    Magnet (number) | schools : Latitude (number) | schools : Longitude (number
    ) | schools : AdmFName1 (text) | schools : AdmLName1 (text) | schools :
    AdmEmail1 (text) | schools : AdmFName2 (text) | schools : AdmLName2 (text)
    | schools : AdmEmail2 (text) | schools : AdmFName3 (text) | schools :
    AdmLName3 (text) | schools : AdmEmail3 (text) | schools : LastUpdate (time)
    ;
 [Primary Keys]: frpm : CDSCode | satscores : cds | schools : CDSCode;
 [Foreign Keys]: frpm : CDSCode equals schools : CDSCode | satscores : cds
     equals schools : CDSCode;
  [detailed description of tables and columns]:
Column description of Table "frpm" have the following descriptions:
Column "County Name" of Table "frpm", means "County Code"
Column "Enrollment (K-12)" of Table frpm has value descriptions "commonsense
    evidence:K-12: 1st grade - 12nd grade"
Column "FRPM Count (K-12)" of Table "frpm", means "Free or Reduced Price Meal
    Count (K-12)", has value descriptions "commonsense evidence:eligible FRPM
    rate = FRPM / Enrollment"
;

  [Database values that related with questions]
The column `County Name` in Table `frpm` has database values: Alameda
The column `cname` in Table `satscores` has database values: Alameda
The column `County` in Table `schools` has database values: Alameda
The column `City` in Table `schools` has database values: Alameda
The column `MailCity` in Table `schools` has database values: Alameda
The column `GSoffered` in Table `schools` has database values: K-12
The column `GSserved` in Table `schools` has database values: K-12
The column `AdmFName1` in Table `schools` has database values: Rae
The column `AdmLName1` in Table `schools` has database values: Free
;
 [Additional Info]: Eligible free rate for K-12 = `FRPM Count (K-12)` / `
    Enrollment (K-12)`
  [Q]: What is the highest eligible free rate for K-12 students in the schools
     in Alameda County?;
  [SQL]:
```