Error analysis

▼ question 21:

`"db_id": "california_schools",`

`"question": "In Los Angeles how many schools have more than 500 free meals but less than 700 free or reduced price meals for K-12?",`

`"evidence": "",`

`"SQL": "SELECT COUNT(CDSCode) FROM frpm WHERE `County Name` = 'Los Angeles' AND `Free Meal Count (K-12)` > 500 AND `Free Meal Count (K-12)` < 700"`,

`"difficulty": "simple"`

`"predicted_sql": "SELECT COUNT(*) FROM frpm T1 JOIN schools T2 ON T1.CDSCode = T2.CDSCode WHERE T2.County = 'Los Angeles' AND T1.`Free Meal Count (K-12)` > 500 AND T1.`FRPM Count (K-12)` < 700`

`"analysis": "incorrect GOLD_SQL : column`

`because the question says less than 700 free or reduced price meals for K-12 not free meals for k-12 and the correct column for this is `FRPM Count (K-12)` based on columns descriptions",`

▼ question 25:

`"db_id": "california_schools",`

`"question": "Name schools in Riverside which the average of average math score for SAT is grater than 400, what is the funding type of these schools?",`

`"evidence": "Average of average math = sum(average math scores) / count(schools).",`

`"SQL": "SELECT T1.sname, T2.`Charter Funding Type` FROM satscores AS T1 INNER JOIN frpm AS T2 ON T1.cds = T2.CDSCode WHERE T2.`District Name` LIKE 'Riverside%' GROUP BY T1.sname, T2.`Charter Funding Type` HAVING CAST(SUM(T1.AvgScrMath) AS REAL) / COUNT(T1.cds) > 400",`

`"difficulty": "moderate"`

`"predicted_qsl": SELECT T2.CDSCode, T2.FundingType FROM satscores T1 JOIN schools T2 ON T1.cds = T2.CDSCode WHERE T2.County = 'Riverside' AND (SELECT AVG(T1.AvgScrMath) FROM satscores T1 JOIN schools T2 ON T1.cds = T2.CDSCode WHERE T2.County = 'Riverside' AND T1.AvgScrMath IS NOT NULL) > 400`

`"analysis": vague question not clear what is Riverside (it is a county name so there is only one value Riverside in the database in column county ) how we can do average`

`of avg on what ???`

question 81:

`"db_id": "california_schools"`

`"question": "In which city can you find the school in the state of California with the lowest latitude coordinates and what is its lowest grade? Indicate the school name."`

`"evidence": "State of California refers to state = 'CA'"`

``"SQL": "SELECT T2.City, T1.`Low Grade`, T1.`School Name` FROM frpm AS T1 INNER JOIN schools AS T2 ON T1.CDSCode = T2.CDSCode WHERE T2.State = 'CA' ORDER BY T2.Latitude ASC LIMIT 1"``

`"difficulty": "moderate"`

`"predicted_sql" :` ``SELECT City, School, `Low Grade` FROM schools WHERE State = 'CA'``

`AND Latitude IS NOT NULL ORDER BY Latitude ASC LIMIT 1`

`"analysis" :incorrect table selection`

`No column Low Grade in table schools it is in table frpm`

question 93:

`"db_id": "financial"`

`"question": "How many male customers who are living in North Bohemia have average salary greater than 8000?"`

`"evidence": "Male means that gender = 'M'; A3 refers to region; A11 pertains to average salary."`

`"SQL": "SELECT COUNT(T1.client_id) FROM client AS T1 INNER JOIN district AS T2 ON T1.district_id = T2.district_id WHERE T1.gender = 'M' AND T2.A3 = 'North Bohemia' AND T2.A11 > 8000"`

`"difficulty": "moderate"`

`"predicted_sql" : SELECT COUNT(*) FROM client T1 JOIN district T2 ON T1.district_id = T2.district_id WHERE T1.gender = 'M' AND T2.A3 = 'north Bohemia' AND T2.A11 > 8000`

`"analysis" : Incorrect golden SQL: entity value`

`because there is no value 'North Bohemia' in the dataset there is 'north Bohemia' as used in the predicted SQL`

▼ question 130:

`"db_id": "financial"`

`"question": "How many of the account holders in South Bohemia still do not own credit cards?"`

`"evidence": "A3 contains the region names; South Bohemia is one of region names."`

`"SQL": "SELECT COUNT(T3.account_id) FROM district AS T1 INNER JOIN client AS T2 ON T1.district_id = T2.district_id INNER JOIN disp AS T3 ON T2.client_id = T3.client_id WHERE T1.A3 = 'south Bohemia' AND T3.type != 'OWNER'"`

`"difficulty": "moderate"`

`"predicted_sql" : "SELECT COUNT(DISTINCT T2.account_id) AS account_count FROM district T1 JOIN account T2 ON T1.district_id = T2.district_id LEFT JOIN disp T3 ON T2.account_id = T3.account_id LEFT JOIN card T4 ON T3.disp_id = T4.disp_id WHERE T1.A3 = 'south Bohemia' AND T4.card_id IS NULL;"`

`"analysis" : incorrect table/column selection`

`because we must see the type column from table disp to know if the client is owner of a credit card`

▼ question 164:

`"db_id": "financial"`

`"question": "Who placed the order with the id 32423?"`

`"evidence": ""`

`"SQL": "SELECT T3.client_id FROM `order` AS T1 INNER JOIN account AS T2 ON T1.account_id = T2.account_id INNER JOIN client AS T3 ON T2.district_id = T3.district_id WHERE T1.order_id = 32423"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T3.client_id FROM `order` T1 JOIN disp T2 ON T1.account_id = T2.account_id JOIN client T3 ON T2.client_id = T3.client_id WHERE T1.order_id = 32423;"`

`"analysis" : incorrect table selection`

`instead of using the table account as the golden sql the predicted query uses table disp which is incorrect`

▼ question 176:

`"db_id": "financial"`

`"question": "What is the amount of debt that client number 992 has, and how is this client doing with payments?"`

`"evidence": ""`

`"SQL": "SELECT T3.amount, T3.status FROM client AS T1 INNER JOIN account AS T2 ON T1.district_id = T2.district_id INNER JOIN loan AS T3 ON T2.account_id = T3.account_id WHERE T1.client_id = 992"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T3.amount, T3.status FROM client T1 JOIN disp T2 ON T1.client_id = T2.client_id JOIN loan T3 ON T2.account_id = T3.account_id WHERE T1.client_id = 992 AND T3.status IN ('B', 'D');"`

`"analysis" : incorrect table selection`

`instead of using the table account as the golden sql the predicted query uses table disp which is incorrect`

question 189:

`"db_id": "financial"`

`"question": "Name the account numbers of female clients who are oldest and have lowest average salary?"`

`"evidence": "Female refers to 'F' in the gender; A11 contains information about average salary"`

`"SQL": "SELECT T3.account_id FROM client AS T1 INNER JOIN district AS T2 ON T1.district_id = T2.district_id INNER JOIN account AS T3 ON T2.district_id = T3.district_id WHERE T1.gender = 'F' ORDER BY T1.birth_date ASC, T2.A11 ASC LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T3.account_id FROM client T1 JOIN district T2 ON T1.district_id = T2.district_id JOIN disp T3 ON T1.client_id = T3.client_id WHERE T1.gender = 'F' AND T1.birth_date = (SELECT birth_date FROM client WHERE gender = 'F' ORDER BY birth_date ASC LIMIT 1) AND T2.A11 = (SELECT A11 FROM district WHERE A11 IS NOT NULL ORDER BY A11 ASC LIMIT 1);"`

`"analysis": incorrect table selection`

`instead of using the table account as the gold en sql the predicted query uses table disp which is incorrect.`

▼ question 231:

`"db_id": "toxicology"`

`"question": "Which bond type accounted for the majority of the bonds found in molecule TR018 and state whether or not this molecule is carcinogenic?"`

`"evidence": "TR018 is the molecule id; majority of the bond found refers to MAX(COUNT(bond_type)); label = '+' mean molecules are carcinogenic; label = '-' means molecules are non-carcinogenic"`

`"SQL": "SELECT T.bond_type FROM ( SELECT T1.bond_type, COUNT(T1.molecule_id) FROM bond AS T1  WHERE T1.molecule_id = 'TR018' GROUP BY T1.bond_type ORDER BY COUNT(T1.molecule_id) DESC LIMIT 1 ) AS T"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT T1.bond_type, T2.label FROM bond T1 JOIN molecule T2 ON T1.molecule_id = T2.molecule_id WHERE T1.molecule_id = 'TR018' GROUP BY T1.bond_type ORDER BY COUNT(T1.bond_type) DESC LIMIT 1;"`

`"analysis": Incorrect golden SQL`

`the question states that we must saye if the molecule is carcinogenic so we must include table molecule in which there is column label to know this information`

`only one error in the predicted sql is that in the count we count T1.molecule_id not bond_type`

▼ question 267:

`"db_id": "toxicology"`

`"question": "List down the bond type for molecules from molecule id TR000 to TR050."`

`"evidence": "double bond refers to bond_type = ' = '; single bond refers to bond_type = '-'; triple bond refers to bond_type = '#';"`

`"SQL": "SELECT T2.molecule_id, T2.bond_type FROM molecule AS T1 INNER JOIN bond AS T2 ON T1.molecule_id = T2.molecule_id WHERE T1.molecule_id BETWEEN 'TR000' AND 'TR050'"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T2.bond_type FROM molecule T1 JOIN bond T2 ON T1.molecule_id = T2.molecule_id WHERE T1.molecule_id BETWEEN 'TR000' AND 'TR050';"`

`"analysis": Incomplete SELECT cols`

`the question mentions to output only the bond_type not also molecule_id but to output the result in a clear way we should output also the molecule id`

question 286:

`"db_id": "toxicology"`

`"question": "Among all chemical compounds identified in the database, what percent of compounds form a triple-bond."`

`"evidence": "triple bond refers to bond_type = '#';"`

`"SQL": "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"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT (COUNT(DISTINCT T1.molecule_id) * 100.0 / (SELECT COUNT(DISTINCT molecule_id) FROM bond)) AS percent_triple_bond FROM bond T1 WHERE T1.bond_type = '#';"`

`"analysis": incorrect column molecule_id for count` `we must count on the bonds because the question says chemical compounds that form a triple bond`

question 307:

`"db_id": "toxicology"`

`"question": "Name the atoms' elements that form bond TR000_2_3."`

`"evidence": "element = 'cl' means Chlorine; element = 'c' means Carbon; element = 'h' means Hydrogen; element = 'o' means Oxygen, element = 's' means Sulfur; element = 'n' means Nitrogen, element = 'p' means Phosphorus, element = 'na' means Sodium, element = 'br' means Bromine, element = 'f' means Fluorine; element = 'i' means Iodine; element = 'sn' means Tin; element = 'pb' means Lead; element = 'te' means Tellurium; element = 'ca' means Calcium"`

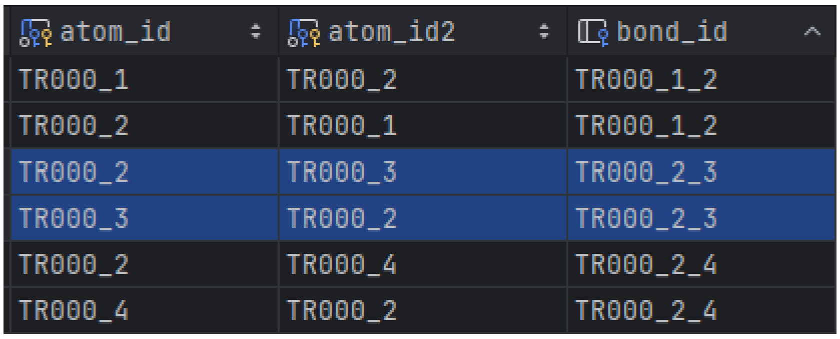
`"SQL": "SELECT T2.element FROM connected AS T1 INNER JOIN atom AS T2 ON T1.atom_id = T2.atom_id WHERE T1.bond_id = 'TR000_2_3'"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT T2.element, T3.element FROM connected T1 JOIN atom T2 ON T1.atom_id = T2.atom_id JOIN atom T3 ON T1.atom_id2 = T3.atom_id WHERE T1.bond_id = 'TR000_2_3';"`

`"analysis": Unnecessary SELECT columns`

` redundant element names because the logic of the prediction is true a bond is between two atoms in this example are 'TR000_2' and 'TR000_3' so in connected table there is 2 atom ids atom_id for first atom and atom_id2 for second atom but in the connected table there is duplicate rows for each bond representing the two reversed orders that's why we will have duplicate results and selecting for only one column atom_id is suffisant.`



atom_id	atom_id2	bond_id
TR000_1	TR000_2	TR000_1_2
TR000_2	TR000_1	TR000_1_2
TR000_2	TR000_3	TR000_2_3
TR000_3	TR000_2	TR000_2_3
TR000_2	TR000_4	TR000_2_4
TR000_4	TR000_2	TR000_2_4

▼ question 326:

`"db_id": "toxicology"`

`"question": "Which molecule consisted of Sulphur atom with double bond?"`

`"evidence": "sulphur refers to element - 's'; double bond refers to bond_type = ' = ';"`

`"SQL": "SELECT DISTINCT T1.molecule_id FROM atom AS T1 INNER JOIN bond AS T2 ON T1.molecule_id = T2.molecule_id WHERE T1.element = 's' AND T2.bond_type = '='"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T1.molecule_id FROM atom T1 JOIN connected T2 ON T1.atom_id = T2.atom_id JOIN bond T3 ON T2.bond_id = T3.bond_id WHERE T1.element = 's' AND T3.bond_type = '=';"`

`"analysis": Unnecessary selected table connected and the non use of DISTINCT clause`

▼ question 338:

`"db_id": "toxicology"`

`"question": "What is the atom ID of double bonded carbon in TR012 molecule?"`

`"evidence": "carbon refers to element = 'c'; double bond refers to bond_type = ' = ';"`

`"SQL": "SELECT T1.atom_id FROM atom AS T1 INNER JOIN molecule AS T2 ON T1.molecule_id = T2.molecule_id INNER JOIN bond AS T3 ON T2.molecule_id = T3.molecule_id WHERE T2.molecule_id = 'TR012' AND T3.bond_type = '=' AND T1.element = 'c'"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.atom_id FROM atom T1 JOIN connected T2 ON T1.atom_id = T2.atom_id JOIN bond T3 ON T2.bond_id = T3.bond_id WHERE T1.element = 'c' AND`

`T3.bond_type = '=' AND T3.molecule_id = 'TR012';"`

`"analysis": Incorrect golden SQL`

`by execution the query doesn’t respond correctly to the expected result where the predicted sql return the estimated result`

▼ question 387:

`"db_id": "card_games"`

`"question": "What are the cards for set OGW? State the colour for these cards."`

`"evidence": "set OGW refers to setCode = 'OGW';"`

`"SQL": "SELECT id, colors FROM cards WHERE id IN ( SELECT id FROM set_translations WHERE setCode = 'OGW' )"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT name, colors FROM cards WHERE setCode = 'OGW';"`

`"analysis": Incorrect golden SQL`

`because table set_translations contains the translation of each setCode into 10 different language each and it's not relevant to answer the posed question and the final answer will be the ids of the translation of setCode 'OGW' and answer will be exactly 10 rows which is incorrect and the predicted SQL is true.`

▼ question 402:

`"db_id": "card_games"`

`"question": "What is the percentage of Story Spotlight cards that also have a text box? List them by their ID."`

`"evidence": "Story Spotlight cards that do not have a text box refers to isStorylight = 1 and isTextless = 1; Percentage refer to DIVIDE(SUM(count(id) where isStorylight = 1), SUM(count(id))) * 100\n\n"`

`"SQL": "SELECT CAST(SUM(CASE WHEN isTextless = 0 THEN 1 ELSE 0 END) AS REAL) * 100 / COUNT(id) FROM cards WHERE isStorySpotlight = 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT id, (100.0 * SUM(CASE WHEN isStorySpotlight = 1 AND isTextless = 0 THEN 1 ELSE 0 END) / SUM(CASE WHEN isStorySpotlight = 1 THEN 1 ELSE 0 END)) AS percentage FROM cards WHERE isStorySpotlight = 1 GROUP BY id;"`

`"analysis" : Incorrect golden SQL`

` because the question states to include also the ID of the cards not only the percentage as in golden sql`

▼ question 423:

`"db_id": "card_games"`

`"question": "Please provide the ids of top three powerful pairs of Kingdom Foil and Kingdom Cards sorted by Kingdom Foil id in alphabetical order."`

`"evidence": "poweful refers to cardKingdomFoilId is not null AND cardKingdomId is not null"`

`"SQL": "SELECT cardKingdomFoilId, cardKingdomId FROM cards WHERE cardKingdomFoilId IS NOT NULL AND cardKingdomId IS NOT NULL ORDER BY cardKingdomFoilId LIMIT 3"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT id FROM cards WHERE cardKingdomFoilId IS NOT NULL AND cardKingdomId IS NOT NULL ORDER BY cardKingdomFoilId ASC LIMIT 3;"`

`"analysis": Incorrect select column`

`the predicted sql returns the id where the question states that we must return the ids cardKingdomFoilId, cardKingdomId (which is not very clear )`

▼ question 448:

`"db_id": "card_games"`

`"question": "Name the foreign name of the card that has abzan watermark? List out the type of this card."`

`"evidence": ""`

`"SQL": "SELECT DISTINCT T1.name, T1.type FROM cards AS T1 INNER JOIN foreign_data AS T2 ON T2.uuid = T1.uuid WHERE T1.watermark = 'abzan'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.name, T2.type FROM cards T1 JOIN foreign_data T2 ON T1.uuid = T2.uuid WHERE T1.watermark = 'abzan';"`

`"analysis": Incorrect golden SQL`

`because the question demands returning the foreign name(from table foreign data) of the card not its name`

▼ question 479:

`"db_id": "card_games"`

`"question": "Among the cards with converted mana cost higher than 5 in the set Coldsnap, how many of them have unknown power?"`

`"evidence": "card set Coldsnap refers to name = 'Coldsnap'; converted mana cost higher than 5 refers to convertedManaCost > 5; unknown power refers to power = '*' or T1.power is null"`

`"SQL": "SELECT SUM(CASE WHEN T1.power LIKE '%*%' OR T1.power IS NULL THEN 1 ELSE 0 END) FROM cards AS T1 INNER JOIN sets AS T2 ON T2.code = T1.setCode WHERE T2.name = 'Coldsnap' AND T1.convertedManaCost > 5"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT COUNT(*) FROM cards T1 JOIN sets T2 ON T1.setCode = T2.code WHERE T2.mcmName = 'Coldsnap' AND T1.convertedManaCost > 5 AND (T1.power = '*' OR T1.power IS NULL);"`

`"analysis": Incorrect golden SQL or evidence` `because in the evidence they said that unknown power refers to power = '*' and in the golden sql they are using LIKE '%*%' not equality`

question 484:

`"db_id": "card_games"`

`"question": "Please list the Italian names of the cards in the set Coldsnap with the highest converted mana cost."`

`"evidence": "card set Coldsnap refers to name = 'Coldsnap'; Italian refers to language = 'Italian'"`

`"SQL": "SELECT T2.name FROM foreign_data AS T1 INNER JOIN cards AS T2 ON T2.uuid = T1.uuid INNER JOIN sets AS T3 ON T3.code = T2.setCode WHERE T3.name = 'Coldsnap' AND T1.language = 'Italian' ORDER BY T2.convertedManaCost DESC LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T3.name FROM sets T1 JOIN cards T2 ON T1.code = T2.setCode JOIN foreign_data T3 ON T2.uuid = T3.uuid WHERE T1.name = 'Coldsnap' AND T3.language = 'Italian' AND T2.convertedManaCost IS NOT NULL ORDER BY T2.convertedManaCost DESC LIMIT 1;"`

`analysis : Incorrect golden SQL`

`because the question says to output the italian name of the card (which is in foreign data table ) and not the original name of the card (from table cards)`

question 515:

`"db_id": "card_games"`

`"question": "When was the oldest mythic card released and what are its legal play formats?"`

`"evidence": "the oldest card refers to MIN(originalReleaseDate); mythic card refers to rarity = 'mythic'; legal play refers to status = 'legal'; play format refers to format"`

`"SQL": "SELECT T1.originalReleaseDate, T2.format FROM cards AS T1 INNER JOIN legalities AS T2 ON T1.uuid = T2.uuid WHERE T1.rarity = 'mythic' AND`

`T1.originalReleaseDate IS NOT NULL AND T2.status = 'Legal' ORDER BY T1.originalReleaseDate LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.originalReleaseDate, T2.format FROM cards T1 JOIN legalities T2 ON T1.uuid = T2.uuid WHERE T1.rarity = 'mythic' AND T2.status = 'Legal' ORDER BY T1.originalReleaseDate ASC LIMIT 1;"`

`analysis: Missing NULL`

`Did not check if the selected column originalReleaseDate has NULL values and exclude them .`

▼ question 520:

`"db_id": "card_games"`

`"question": "Who is the illustrator that illustrated the least amount of cards? List the format of play of the cards that he/she illustrated."`

`"evidence": "format of the cards refers to format; illustrator refers to artist; the least amount of cards refers to MIN(artist)"`

`"SQL": "SELECT T1.artist, T2.format FROM cards AS T1 INNER JOIN legalities AS T2 ON T2.uuid = T1.uuid GROUP BY T1.artist ORDER BY COUNT(T1.id) ASC LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "WITH LeastIllustratedArtist AS (SELECT artist FROM cards GROUP BY artist ORDER BY COUNT(*) ASC LIMIT 1) SELECT l.format FROM legalities l JOIN cards c ON l.uuid = c.uuid WHERE c.artist = (SELECT artist FROM LeastIllustratedArtist);"`

`analysis: JOIN`

`there is an error because the join of tables must be before selecting the min`

`illustrated artist join and then select not select artist and then format`

▼ question 529:

`"db_id": "card_games"`

`"question": "Find and list the names of sets which doesn't have Japanese translation but have Korean translation."`

`"evidence": "names of sets refers to name; doesn't have Japanese translation refers to language not like '%Japanese%'; have Korean translation refers to language = 'Korean'"`

`"SQL": "SELECT name FROM sets WHERE code IN ( SELECT setCode FROM set_translations WHERE language = 'Korean' AND language NOT LIKE '%Japanese%' )"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T2.name FROM sets T2 WHERE T2.code IN (SELECT T1.setCode FROM set_translations T1 WHERE T1.language = 'Korean' ) AND T2.code NOT IN (SELECT T1.setCode FROM set_translations T1 WHERE T1.language LIKE '%Japanese%')"`

`analysis : incorrect golden sql`

`note that also the predicted sql is not entirely correct it misses small checks`

`to check if a set has a translation in some language we must check that translation column when selecting this language is not null`.

`correct sql :` `SELECT T2.name FROM sets T2 WHERE T2.code IN (SELECT T1.setCode FROM set_translations T1 WHERE T1.language = 'Korean' AND T1.translation IS NOT NULL ) AND T2.code NOT IN (SELECT T1.setCode FROM set_translations T1 WHERE T1.language LIKE '%Japanese%' AND T1.translation IS NOT NULL)`

▼ question 538:

`"db_id": "codebase_community"`

`"question": "Please list the titles of the posts owned by the user csgillespie?"`

`"evidence": "\"csgillespie\" is the DisplayName of user"`

`"SQL": "SELECT T1.Title FROM posts AS T1 INNER JOIN users AS T2 ON T1.OwnerUserId = T2.Id WHERE T2.DisplayName = 'csgillespie'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.Title FROM users T1 JOIN posts T2 ON T1.Id = T2.OwnerUserId WHERE T1.DisplayName = 'csgillespie' AND T2.Title IS NOT NULL;"`

`analysis: unnecessary NULL check`

`the predicted sql in candidate generation is correct but after revision it adds the unnecessary NULL check for title`

▼ question 586:

`"db_id": "codebase_community"`

`"question": "Which user added a bounty amount of 50 to the post title mentioning variance?"`

`"evidence": "\"bounty amount of 50 refers to BountyAmount = 50; user refers to DisplayName"`

`"SQL": "SELECT T3.DisplayName, T1.Title FROM posts AS T1 INNER JOIN votes AS T2 ON T1.Id = T2.PostId INNER JOIN users AS T3 ON T3.Id = T2.UserId WHERE T2.BountyAmount = 50 AND T1.Title LIKE '%variance%'"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT T3.DisplayName FROM votes T1 JOIN posts T2 ON T1.PostId = T2.Id JOIN users T3 ON T1.UserId = T3.Id WHERE T1.BountyAmount = 50 AND T2.Title LIKE '%variance%'"`

`analysis : Incorrect golden SQL`

`not demanded in the question to include the post title in the response just the user`

`name so the predicted sql is correct and the golden sql have unnecessary column`

▼ question 593:

`"db_id": "codebase_community"`

`"question": "How many users from New York have a teacher and supporter badge?"`

`"evidence": "\"Supporter\" and \"Teachers\" are both Name of badge; 'New York' is the Location; user refers to UserId"`

`"SQL": "SELECT COUNT(DISTINCT T1.Id) FROM badges AS T1 INNER JOIN users AS T2 ON T1.UserId = T2.Id WHERE T1.Name IN ('Supporter', 'Teacher') AND T2.Location = 'New York'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT COUNT(DISTINCT T1.Id) FROM users T1 JOIN badges T2 ON T1.Id = T2.UserId WHERE T1.Location = 'New York' AND T2.Name IN ('Supporter', 'Teacher') GROUP BY T1.Id HAVING COUNT(DISTINCT T2.Name) = 2;"`

`analysis: Unnecessary group by clause`

`the predicted sql have added an unnecessary group by clause for the query`

`and the golden sql has count for badge id which is incorrect it must have count for user id`

`correct sql : SELECT COUNT(DISTINCT T1.Id) FROM users T1 JOIN badges T2 ON T1.Id =`

`T2.UserId WHERE T1.Location = 'New York' AND T2.Name IN ('Supporter', 'Teacher')`

▼ question 600:

`"db_id": "codebase_community"`

`"question": "List out all post that are related to post ID 61217 and what is the popularity of this post?"`

`"evidence": "post related refers to RelatedPostId; popularity refers to ViewCount"`

`"SQL": "SELECT T1.ViewCount FROM posts AS T1 INNER JOIN postLinks AS T2 ON T1.Id = T2.PostId WHERE T2.PostId = 61217"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T1.PostId, T2.ViewCount FROM postLinks T1 JOIN posts T2 ON T1.PostId = T2.Id WHERE T1.RelatedPostId = 61217 UNION SELECT 61217 AS PostId, T3.ViewCount FROM posts T3 WHERE T3.Id = 61217;"`

`analysis : incorrect golden SQL`

`the golden sql doesn’t respond correctly to the question but the predicted query is correct`

▼ question 617:

`"db_id": "codebase_community"`

`"question": "What is the detailed content of the comment of the post which was created on 7/19/2010 7:37:33 PM?"`

`"evidence": "detailed content of the comment refers to Text; created on 7/19/2010 7:37:33 PM CreationDate = 2010-07-19 19:37:33.0'"`

`"SQL": "SELECT T1.Text FROM comments AS T1 INNER JOIN posts AS T2 ON T1.PostId = T2.Id WHERE T1.CreationDate = '2010-07-19 19:37:33.0'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.Text FROM posts T1 JOIN comments T2 ON T1.Id = T2.PostId WHERE T1.CreaionDate = '2010-07-19 19:37:33.0';"`

`analysis: incorrect column CreationDate`

`the predicted sql uses creationdate from the table posts however it must use the creationdate from table comments as the golden sql`

▼ question 631:

`"db_id": "codebase_community"`

`"question": "How many posts were created by Daniel Vassallo?"`

`"evidence": "DisplayName = 'Daniel Vassallo';"`

`"SQL": "SELECT COUNT(T1.Id) FROM users AS T1 INNER JOIN postHistory AS T2 ON T1.Id = T2.UserId WHERE T1.DisplayName = 'Daniel Vassallo'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT COUNT(*) FROM posts T1 JOIN users T2 ON T1.OwnerUserId = T2.Id WHERE T2.DisplayName = 'Daniel Vassallo';"`

`"analysis": incorrect table selection`

`selecting from table posts instead of using table postHistory`

question 656:

`"db_id": "codebase_community"`

`"question": "Describe the display name of the parent ID for child post with the highest score."`

`"evidence": "If the parent id is not null, the post is the child post; the highest score refers to MAX(Score);"`

`"SQL": "SELECT DisplayName FROM users WHERE Id = ( SELECT OwnerUserId FROM posts WHERE ParentId IS NOT NULL ORDER BY Score DESC LIMIT 1 )"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T3.DisplayName FROM posts T1 JOIN posts T2 ON T1.ParentId = T2.Id JOIN users T3 ON T2.OwnerUserId = T3.Id WHERE T1.ParentId IS NOT NULL AND T1.Score IS NOT NULL ORDER BY T1.Score DESC LIMIT 1;"`

`analysis: Incorrect golden SQL`

`the golden sql doesn't respond correctly to the question however the predicted query do.`

question 729:

`"db_id": "superhero"`

`"question": "What is the average height of the superheroes from Marvel Comics?"`

`"evidence": "superheroes from Marvel Comics refers to publisher_name = 'Marvel Comics'; average height of the superheroes refers to AVG(height_cm)"`

`"SQL": "SELECT AVG(T1.height_cm) FROM superhero AS T1 INNER JOIN publisher AS T2 ON T1.publisher_id = T2.id WHERE T2.publisher_name = 'Marvel Comics'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT AVG(T1.height_cm) AS average_height FROM superhero T1 JOIN publisher T2 ON T1.publisher_id = T2.id WHERE T2.publisher_name = 'Marvel Comics' AND T1.height_cm IS NOT NULL AND T1.height_cm != 0;"`

`analysis : Unnecessary check for height_cm != 0`

▼ question 772:

`"db_id": "superhero"`

`"question": "List the eyes, hair and skin colour of all female superheroes published by Dark Horse Comics."`

`"evidence": "eyes refers to eye_colour_id; hair refers to hair_colour_id; skin colour refers to skin_colour_id; female superheroes refers to gender = 'Female'; published by Dark Horse Comics refers to publisher_name = 'Dark Horse Comics';"`

`"SQL": "SELECT T1.eye_colour_id, T1.hair_colour_id, T1.skin_colour_id FROM superhero AS T1 INNER JOIN publisher AS T2 ON T2.id = T1.publisher_id INNER JOIN gender AS T3 ON T3.id = T1.gender_id WHERE T2.publisher_name = 'Dark Horse Comics' AND T3.gender = 'Female'"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT T4.colour AS eye_colour, T5.colour AS hair_colour, T6.colour AS skin_colour FROM superhero T1 JOIN gender T2 ON T1.gender_id = T2.id JOIN publisher T3 ON T1.publisher_id = T3.id JOIN colour T4 ON T1.eye_colour_id = T4.id JOIN colour T5 ON T1.hair_colour_id = T5.id JOIN colour T6 ON T1.skin_colour_id = T6.id WHERE T2.gender = 'Female' AND T3.publisher_name = 'Dark Horse Comics';"`

`analysis : it is correct`

`mainly the 2 queries have same result but the predicted query present the colour by its name which is more clear than presenting only the id in the golden sql`

question 812:

`"db_id": "superhero"`

`"question": "List down at least five full names of superheroes with blue eyes."`

`"evidence": "blue eyes refers to colour.colour = 'Blue' WHERE eye_colour_id = colour.id;"`

`"SQL": "SELECT T1.superhero_name FROM superhero AS T1 INNER JOIN colour AS T2 ON T1.eye_colour_id = T2.id WHERE T2.colour = 'Blue' LIMIT 5"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T1.full_name FROM superhero T1 JOIN colour T2 ON T1.eye_colour_id = T2.id WHERE T2.colour = 'Blue' LIMIT 5;"`

`analysis : Incorrect golden SQL`

`the question demands the full names of superheroes not their names so the predict sql is correct and the golden sql is not correct`

question 857:

`"db_id": "formula_1"`

`"question": "Give the coordinate position for Abu Dhabi Grand Prix."`

`"evidence": "coordinates refers to (lat, lng); position and location shares the same meaning."`

`"SQL": "SELECT DISTINCT T1.lat, T1.lng, T1.location FROM circuits AS T1 INNER JOIN races AS T2 ON T2.circuitID = T1.circuitId WHERE T2.name = 'Abu Dhabi Grand Prix'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.lat, T2.lng FROM races T1 JOIN circuits T2 ON T1.circuitId = T2.circuitId WHERE T1.name = 'Abu Dhabi Grand Prix';"`

`analysis : Missing DISTINCT clause in the predicted SQL and also unnecessary location column in the golden sql as the question demands only coordinate position`

▼ question 860:

`"db_id": "formula_1"`

`"question": "For the driver who had the Q2 time as 0:01:40 in the qualifying race No. 355, what is his nationality?"`

`"evidence": "race number refers to raceId;"`

`"SQL": "SELECT DISTINCT T2.nationality FROM qualifying AS T1 INNER JOIN drivers AS T2 ON T2.driverId = T1.driverId WHERE T1.raceId = 355 AND T1.q2 LIKE '1:40%'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.nationality FROM qualifying T1 JOIN drivers T2 ON T1.driverId = T2.driverId WHERE T1.q2 = '1:40.473' AND T1.raceId = 355;"`

`analysis: wrong entity in where clause '1:40.473'(because of entity retrieval) and also missing DISTINCT clause`

`Note that the question is vague because it doesn't specify exactly the time because time is in format m:s.ms not only m:s`

▼ question 896:

`"db_id": "formula_1"`

`"question": "Calculate the percentage whereby Hamilton was not at the 1st track of the the f1 circuit since 2010."`

`"evidence": "DIVIDE(COUNT(raceId) where surname = 'Hamilton', year >= 2010 and position>1), (COUNT(raceId) where surname = 'Hamilton', year >= 2010) as percentage;"`

`"SQL": "SELECT CAST(COUNT(CASE WHEN T2.position <> 1 THEN T2.position END) AS REAL) * 100 / COUNT(T2.driverStandingsId) FROM races AS T1 INNER JOIN driverStandings AS T2 ON T2.raceId = T1.raceId INNER JOIN drivers AS T3 ON T3.driverId = T2.driverId WHERE T3.surname = 'Hamilton' AND T1.year >= 2010"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT 100.0 * (COUNT(CASE WHEN T3.position > 1 THEN 1 END) * 1.0 / COUNT(T3.raceId)) AS percentage FROM drivers T1 JOIN results T3 ON T1.driverId = T3.driverId JOIN races T2 ON T2.raceId = T3.raceId WHERE T1.surname = 'Hamilton' AND T2.year >= 2010;"`

`analysis : Incorrect table selection`

`instead of using table driverStandings as golden sql the predicted sql uses table results`

`Note that it is also correct to use results table because it stores races results but what is the most correct to use from those 2 tables is not clear`

question 929:

`"db_id": "formula_1"`

`"question": "Please list the Formula_1 races that Lewis Hamilton participated."`

`"evidence": ""`

`"SQL": "SELECT T1.name FROM races AS T1 INNER JOIN results AS T2 ON T2.raceId = T1.raceId INNER JOIN drivers AS T3 ON T3.driverId = T2.driverId WHERE T3.forename = 'Lewis' AND T3.surname = 'Hamilton'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T3.raceId FROM drivers T1 JOIN results T2 ON T1.driverId = T2.driverId JOIN races T3 ON T2.raceId = T3.raceId WHERE T1.forename = 'Lewis' AND T1.surname = 'Hamilton';"`

`analysis : Incorrect output column raceId`

`note that it is not specified in the question to output race name but it is more readable to include their names instead of id .However the predicted sql is not incorrect also`

question 937:

`"db_id": "formula_1"`

`"question": "What's the finish time for the driver who ranked second in 2008's Australian Grand Prix?"`

`"evidence": "finish time refers to time"`

`"SQL": "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"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.time FROM races T1 JOIN results T2 ON T1.raceId = T2.raceId WHERE T1.year = 2008 AND T1.name = 'Australian Grand Prix' AND T2.positionOrder = 2;"`

`analysis : Incorrect golden SQL`

`because that rank column from results table states for "starting rank positioned by fastest lap speed" but the column positionOrder states for "the finishing order of positions " so the latter is the correct column to use`

▼ question 944:

`"db_id": "formula_1"`

`"question": "How much faster in percentage is the champion than the driver who finished the race last in the 2008 Australian Grand Prix?"`

`"evidence": "how much faster in percentage = divide(subtract(incremental time, champion time), last_driver time) * 100%; last driver finished time = incremental time + champion time; only champion's finished time is represented by 'HH:MM:SS.mmm'; finished the game refers to time is not null"`

`"SQL": "WITH time_in_seconds AS (SELECT T1.positionOrder, CASE WHEN T1.positionOrder = 1 THEN (CAST(SUBSTR(T1.time, 1, 1) AS REAL) * 3600) + (CAST(SUBSTR(T1.time, 3, 2) AS REAL) * 60) + CAST(SUBSTR(T1.time, 6) AS REAL) ELSE CAST(SUBSTR(T1.time, 2) AS REAL) END AS time_seconds FROM results AS T1 INNER JOIN races AS T2 ON T1.raceId = T2.raceId WHERE T2.name = 'Australian Grand Prix' AND T1.time IS NOT NULL AND T2.year = 2008), champion_time AS (SELECT time_seconds FROM time_in_seconds WHERE positionOrder = 1), last_driver_incremental AS (SELECT time_seconds FROM time_in_seconds WHERE positionOrder = (SELECT MAX(positionOrder) FROM time_in_seconds)) SELECT (CAST((SELECT time_seconds FROM last_driver_incremental) AS REAL) * 100) / (SELECT` `time_seconds + (SELECT time_seconds FROM last_driver_incremental) FROM champion_time)"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT ((CAST(SUBSTR(T2.time, 1, INSTR(T2.time, ':') - 1) AS INTEGER) * 60 + CAST(SUBSTR(T2.time, INSTR(T2.time, ':') + 1) AS REAL)) / (CAST(SUBSTR(T2.time, 1, INSTR(T2.time, ':') - 1) AS INTEGER) * 60 + CAST(SUBSTR(T2.time, INSTR(T2.time, ':') + 1) AS REAL) + CAST(T3.time AS REAL)) * 100) AS percentage_faster FROM races T1 JOIN results T2 ON T1.raceId = T2.raceId JOIN results T3 ON T1.raceId = T3.raceId WHERE T1.year = 2008 AND T1.name = 'Australian Grand Prix' AND T2.positionOrder = 1 AND T3.positionOrder = (SELECT MAX(positionOrder) FROM results WHERE raceId = T1.raceId AND time IS NOT NULL);"`

`analysis : it is a hard query but using the execution results seems that the predicted query is more logical than the golden sql`

question 956:

`"db_id": "formula_1"`

`"question": "Which drivers born after 1975 have been ranked 2? Please give their forenames and surnames."`

`"evidence": "born after 1975 refers to year(dob) >1975;"`

`"SQL": "SELECT T2.forename, T2.surname FROM results AS T1 INNER JOIN drivers AS T2 on T1.driverId = T2.driverId WHERE STRFTIME('%Y', T2.dob) > '1975' AND T1.rank = 2"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T1.forename, T1.surname FROM drivers T1 JOIN driverStandings T2 ON T1.driverId = T2.driverId WHERE strftime('%Y', T1.dob) > '1975' AND T2.position = 2;"`

`analysis : Incorrect golden SQL`

`because as we said in latter example the rank column doesn't mean the final rank in the race but the starting rank and position column is the correct column`

▼ question 972:

`"db_id": "formula_1"`

`"question": "Which drivers who were born in 1971 and has the fastest lap time on the race? Give id and code of these drivers."`

`"evidence": "born in 1971 refers to year(dob) = 1971; has the fastest lap time refers to fastestLapTime has values"`

`"SQL": "SELECT T2.driverId, T2.code FROM results AS T1 INNER JOIN drivers AS T2 on T1.driverId = T2.driverId WHERE STRFTIME('%Y', T2.dob) = '1971' AND T1.fastestLapTime IS NOT NULL"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.driverId, T1.code FROM drivers T1 JOIN results T2 ON T1.driverId = T2.driverId WHERE strftime('%Y', T1.dob) = '1971' AND T2.fastestLapTime IS NOT NULL ORDER BY T2.fastestLapTime ASC LIMIT 1;"`

`analysis : Unnecessary ORDER BY clause`

`we need to output all drivers that have a fastest lap time in a race, not only the best one of them.`

▼ question 996:

`"db_id": "formula_1"`

`"question": "What is the annual average number of races held during the first 10 years of the 21st century?"`

`"evidence": "races in date between '2000-01-01' and '2010-12-31'"`

`"SQL": "SELECT CAST(SUM(CASE WHEN year BETWEEN 2000 AND 2010 THEN 1 ELSE 0 END) AS REAL) / 10 FROM races WHERE date BETWEEN '2000-01-01' AND '2010-12-31'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT AVG(race_count) AS annual_average_races FROM (SELECT COUNT(raceId) AS race_count FROM races WHERE date BETWEEN '2000-01-01' AND '2010-12-`

`31' GROUP BY year);"`

`analysis : Vague Question / False evidence / Incorrect golden SQL`

`because the first 10 years of 21st century are from 2000 to 2009 and from 2000 to 2010 it is 11 years so the average must be divided by 11. if we go with the evidence or we count year between 2000 and 2009 if we will go with question`

▼ question 1040:

`"db_id": "european_football_2"`

`"question": "List the top 10 players' names whose heights are above 180 in descending order of average heading accuracy."`

`"evidence": "height > 180; average heading accuracy = DIVIDE(SUM(heading_accuracy), COUNT(player_fifa_api_id));"`

```
"SQL": "SELECT t1.player_name FROM Player AS t1 INNER JOIN Player_Attributes AS t2 ON t1.player_api_id = t2.player_api_id WHERE t1.height > 180 GROUP BY t1.id ORDER BY CAST(SUM(t2.heading_accuracy) AS REAL) / COUNT(t2.`player_fifa_api_id`) DESC LIMIT 10"
```

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.player_name FROM Player T1 JOIN Player_Attributes T2 ON T1.player_api_id = T2.player_api_id WHERE T1.height > 180 GROUP BY T1.player_name ORDER BY AVG(T2.heading_accuracy) DESC LIMIT 10;"`

`analysis : Incorrect column in GROUP BY clause`

`in the predicted sql the group by is by player_name column (which is not unique there is players with same name ) so we must use the unique id in the group by as in the golden sql`

▼ question 1085:

`"db_id": "european_football_2"`

`"question": "Which of these players performs the best in crossing actions, Alexis, Ariel Borysiuk or Arouna Kone?"`

`"evidence": "player who perform best in crossing actions refers to MAX(crossing);"`

`"SQL": "SELECT t1.player_name, t2.crossing FROM Player AS t1 INNER JOIN Player_Attributes AS t2 ON t1.player_api_id = t2.player_api_id WHERE t1.player_name IN ('Alexis', 'Ariel Borysiuk', 'Arouna Kone') ORDER BY t2.crossing DESC LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.player_api_id FROM Player T1 JOIN Player_Attributes T2 ON T1.player_api_id = T2.player_api_id WHERE T1.player_name IN ('Alexis', 'Ariel Borysiuk', 'Arouna Kone') ORDER BY T2.crossing DESC LIMIT 1;"`

`analysis : Incorrect output column`

`the question doesn't mention to return names or id of the player but in the question provides the list of players by their names so logically the result must be the name of the player not its id`

`Note that it unnecessary to add column crossing in the result !`

question 1087:

`"db_id": "european_football_2"`

`"question": "Among the players whose height is over 180, how many of them have a volley score of over 70?"`

`"evidence": "height > 180; volley score refers to volleys; volleys > 70;"`

`"SQL": "SELECT COUNT(DISTINCT t1.id) FROM Player AS t1 INNER JOIN Player_Attributes AS t2 ON t1.player_api_id = t2.player_api_id WHERE t1.height > 180 AND t2.volleys > 70"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT COUNT(*) FROM Player T1 JOIN Player_Attributes T2 ON T1.player_api_id = T2.player_api_id WHERE T1.height > 180 AND T2.volleys > 70;"`

`analysis : absence of DISTINCT clause`

`because the players have many rows in the Player_Attributes table so it will be for sure duplicate rows for same player of we don't apply DISTINCT.`

question 1213:

`"db_id": "thrombosis_prediction"`

`"question": "Name the ID of the patient who is born on the April 1st, 1982. Is his/her alkaliphophatase (ALP) within normal range?"`

`"evidence": "alkaliphophatase (ALP) within normal range refers to ALP < 300"`

`"SQL": "SELECT T1.ID , CASE WHEN T2.ALP < 300 THEN 'normal' ELSE 'abNormal' END FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.Birthday = '1982-04-01'"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.ID, T2.ALP < 300 AS is_ALP_within_normal_range FROM Patient T1 JOIN Laboratory T2 ON T1.ID = T2.ID WHERE T1.Birthday = '1982-04-01' AND T2.ALP IS NOT NULL;"`

`analysis : Correct golden sql and predicted sql`

`the question doesn't specify how to output the information about ALP so it is correct for the 2 cases`

question 1217:

`"db_id": "thrombosis_prediction"`

`"question": "For all patient born in 1982, state if their albumin is within normal range."`

`"evidence": "Year(Birthday) = '1982'; albumin is within normal range refers to ALB between 3.5 and 5.5"`

`"SQL": "SELECT CASE WHEN T2.ALB >= 3.5 AND T2.ALB <= 5.5 THEN 'normal' ELSE 'abnormal' END FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE STRFTIME('%Y', T1.Birthday) = '1982'"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.ID, (T2.ALB > 3.5 AND T2.ALB < 5.5) AS is_albumin_normal FROM Patient T1 JOIN Laboratory T2 ON T1.ID = T2.ID WHERE strftime('%Y', T1.Birthday) = '1982' AND T2.ALB IS NOT NULL;"`

`analysis : same as latter question it is in the format of how to output information of normal abnormal patient`

▼ question 1224:

`"db_id": "thrombosis_prediction"`

`"question": "What is the highest total bilirubin level recorded? List out the patient details with ID, sex and birthday with that index."`

`"evidence": "the highest total bilirubin refers to MAX(T-BIL)"`

`` "SQL": "SELECT T2.`T-BIL`, T1.ID, T1.SEX, T1.Birthday FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID ORDER BY T2.`T-BIL` DESC LIMIT 1" ``

`"difficulty": "simple"`

`` "Predicted SQL": "SELECT T2.ID, T2.SEX, T2.Birthday FROM Laboratory T1 JOIN Patient T2 ON T1.ID = T2.ID WHERE T1.`T-BIL` IS NOT NULL ORDER BY T1.`T-BIL` DESC LIMIT 1;" ``

`analysis : Incomplete column output`

`the predicted query must also include the T-BIL column it is demanded in the question`

▼ question 1247:

`"db_id": "thrombosis_prediction"`

`"question": "Among the male patients who have a normal level of white blood cells, how many of them have an abnormal fibrinogen level?"`

`"evidence": "male patients refers to Sex = 'M'; normal level of white blood cells refers to WBC between 3.5 and 9.0; abnormal fibrinogen level refers to FG < = 150 or FG > = 450; Don't compute repetitive ones."`

`"SQL": "SELECT COUNT(DISTINCT T1.ID) FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T2.FG <= 150 OR T2.FG >= 450 AND T2.WBC > 3.5 AND T2.WBC < 9.0 AND T1.SEX = 'M'"`

`"difficulty": "challenging"`

`"Predicted SQL": "SELECT COUNT(DISTINCT T1.ID) FROM Patient T1 JOIN Laboratory T2 ON T1.ID = T2.ID WHERE T1.SEX = 'M' AND T2.WBC > 3.5 AND T2.WBC < 9.0 AND (T2.FG <= 150 OR T2.FG >= 450);"`

`analysis : Incorrect golden SQL`

`missing parentheses in the clause T2.FG <= 150 OR T2.FG >= 450 because and operator has more priority than or so the clause will be not correct if we don't put the parentheses.`

question 1268:

`db_id": "thrombosis_prediction"`

`"question": "For the patients with an abnormal anti-SM, please list the IDs of the three youngest ones."`

`"evidence": "abnormal anti-SM refers to SM NOT IN('-', '+-'); youngest refers to MAX(Birthday);"`

`"SQL": "SELECT T1.ID FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T2.SM NOT IN ('negative','0') ORDER BY T1.Birthday DESC LIMIT 3"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.ID FROM Laboratory T1 JOIN Patient T2 ON T1.ID = T2.ID WHERE T1.SM NOT IN ('-', '+-') ORDER BY T2.Birthday DESC LIMIT 3;"`

`analysis : Incorrect entities / Incorrect evidence`

`there is no values in the SM column that equals '-' or '+-' we must use values 'negative' and '0' instead as the golden sql but there is no information for that in the column description or any other place and the evidence says to use symbols '-' and '+-'`

question 1272:

`"db_id": "thrombosis_prediction"`

`"question": "Which patient is the first patient with an abnormal anti-SSA to come to the hospital? Please give his or her ID."`

`"evidence": "first patient refers to ID with MIN(`First Date`); abnormal anti-SSA refers to SSA NOT IN('-', '+-');"`

`"SQL": "SELECT T1.ID FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.`First Date` IS NOT NULL AND T2.SSA NOT IN ('negative', '0') ORDER BY T1.`First Date` ASC LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.ID FROM Patient T1 JOIN Laboratory T2 ON T1.ID = T2.ID WHERE T2.SSA NOT IN ('-', '+-') ORDER BY T1.`First Date` LIMIT 1;"`

`analysis :Incorrect entities / Incorrect evidence`

`there is no values in the SM column that equals '-' or '+-' we must use values 'negative' and '0' instead as the golden sql but there is no information for that in the column description or any other place and the evidence says to use symbols '-' and '+-'`

question 1291:

`"db_id": "thrombosis_prediction"`

`"question": "How many male patients have a normal level of both albumin and total protein?"`

`"evidence": "male refers to Sex = 'M'; normal level of both albumin and total protein refers to ALB > 3.5 and ALB < 5.5 AND TP between 6.0 and 8.5;"`

`"SQL": "SELECT COUNT(T1.ID) FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.SEX = 'M' AND T2.ALB BETWEEN 3.5 AND 5.5 AND T2.TP BETWEEN 6.0 AND 8.5"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT COUNT(DISTINCT T1.ID) FROM Patient T1 JOIN Laboratory T2 ON T1.ID = T2.ID WHERE T1.SEX = 'M' AND T2.ALB > 3.5 AND T2.ALB < 5.5 AND T2.TP > 6.0 AND T2.TP < 8.5;"`

`analysis : Incorrect golden SQL`

`absence of DISTINCT clause in the count in the golden sql and the results have many duplicate patients (because table laboratory present the different analysis so it is logic for a patient to have many rows as the number of analysis that have)`

question 1453:

`"db_id": "student_club"`

`"question": "List the name of events with less than average parking cost."`

`"evidence": "name of events refers to event_name; less than average parking cost refers to cost < DIVIDE(SUM(cost), COUNT(event_id)) where category = 'Parking'"`

`"SQL": "SELECT T1.event_name FROM event AS T1 INNER JOIN budget AS T2 ON T1.event_id = T2.link_to_event INNER JOIN expense AS T3 ON T2.budget_id = T3.link_to_budget WHERE T2.category = 'Parking' AND T3.cost < (SELECT AVG(cost) FROM expense)"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T3.event_name FROM event T3 JOIN budget T2 ON T3.event_id = T2.link_to_event JOIN expense T1 ON T2.budget_id = T1.link_to_budget WHERE T1.expense_description = 'Parking' AND T1.cost < (SELECT SUM(T1.cost) / COUNT(T3.event_id) FROM budget T2 JOIN expense T1 ON T2.budget_id = T1.link_to_budget WHERE T1.expense_description = 'Parking')"`

`analysis : predicted SQL has syntax error`

`SQL error or missing database (misuse of aggregate: COUNT(T3.event_id))`

question 1454:

`"db_id": "student_club"`

`"question": "What is the percentage of the cost for the game events?"`

`"evidence": "game events refers to type = 'Game'; percentage = DIVIDE( SUM(cost), COUNT(event_id)) * 100"`

`"SQL": "SELECT SUM(CASE WHEN T1.type = 'Game' THEN T3.cost ELSE 0 END) * 100 / SUM(T3.cost) FROM event AS T1 INNER JOIN budget AS T2 ON T1.event_id = T2.link_to_event INNER JOIN expense AS T3 ON T2.budget_id = T3.link_to_budget"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT (SUM(T2.cost) / COUNT(T1.event_id)) * 100 AS percentage FROM event T1 JOIN expense T2 ON T1.event_id = T2.link_to_budget WHERE T1.type = 'Game' AND T2.cost IS NOT NULL;"`

`analysis : Incorrect column in COUNT`

`due to Incorrect evidence`

`the evidence states that percentage = DIVIDE( SUM(cost), COUNT(event_id)) * 100 where it is false the percentage must be calculated as sum cost of game event divided by total cost of all events.`

question 1466:

`"db_id": "student_club"`

`"question": "Write the full name of the club member with the position of 'Secretary' and list which college the club member belongs to."`

`"evidence": "full name refers to first_name, last name"`

`"SQL": "SELECT T1.first_name, T1.last_name, college FROM member AS T1 INNER JOIN major AS T2 ON T2.major_id = T1.link_to_major WHERE T1.position = 'Secretary'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T1.first_name, T1.last_name, T3.college FROM member T1 JOIN zip_code T2 ON T1.zip = T2.zip_code JOIN major T3 ON T1.link_to_major = T3.major_id WHERE T2.city = 'Secretary';"`

`analysis : Incorrect table selection`

`unnecessary join table zip_code`

`note that the candidate query is correct this error is after revision module`

question 1467:

`"db_id": "student_club"`

`"question": "Calculate the total amount spent on speaker gifts and list the name of the event they were spent on."`

`"evidence": "total amount spent = SUM(spent) where category = 'Speaker Gifts'"`

`"SQL": "SELECT SUM(T1.spent), T2.event_name FROM budget AS T1 INNER JOIN event AS T2 ON T1.link_to_event = T2.event_id WHERE T1.category = 'Speaker Gifts'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T2.event_name, SUM(T1.spent) AS total_spent FROM budget T1 JOIN event T2 ON T1.link_to_event = T2.event_id WHERE T1.category = 'Speaker Gifts' GROUP BY T2.event_name;"`

`analysis: Incorrect golden SQL`

`the result of golden sql is not correct as there not only one event that spent on speaker gifts so we must group by event name the result`

▼ question 1472:

`"db_id": "debit_card_specializing"`

`"question": "In 2012, who had the least consumption in LAM?"`

`"evidence": "Year 2012 can be presented as Between 201201 And 201212, which means between January and December in 2012"`

`"SQL": "SELECT T1.CustomerID FROM customers AS T1 INNER JOIN yearmonth AS T2 ON T1.CustomerID = T2.CustomerID WHERE T1.Segment = 'LAM' AND T2.date BETWEEN 201201 AND 201212 GROUP BY T1.CustomerID ORDER BY SUM(T2.Consumption) ASC LIMIT 1"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T1.CustomerID FROM customers T1 JOIN yearmonth T2 ON T1.CustomerID = T2.CustomerID WHERE T2.Date BETWEEN '201201' AND '201212' AND T1.Segment = 'LAM' ORDER BY T2.Consumption ASC LIMIT 1;"`

`analysis : Missing GROUP BY`

`the predicted sql misses group by clause which is incorrect because the table have many rows with same customerID`

▼ question 1511:

`"db_id": "debit_card_specializing"`

`"question": "For the customers who paid in the euro, what is their average total price of the transactions?"`

`"evidence": ""`

`"SQL": "SELECT AVG(T1.Price) FROM transactions_1k AS T1 INNER JOIN gasstations AS T2 ON T1.GasStationID = T2.GasStationID INNER JOIN customers AS T3 ON T1.CustomerID = T3.CustomerID WHERE T3.Currency = 'EUR'"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT AVG(T1.Amount * T1.Price) AS average_total_price FROM transactions_1k T1 JOIN customers T2 ON T1.CustomerID = T2.CustomerID WHERE T2.Currency = 'euro' AND T1.Amount IS NOT NULL AND T1.Price IS NOT NULL;"`

`analysis : Incorrect entity`

`the value 'euro' is not present in the table we must search for 'EUR' for currency euro`

`note that the golden sql is incorrect because the total price is price * amount`

▼ question 1527:

`"db_id": "debit_card_specializing"`

`"question": "Which gas station has the highest amount of revenue?"`

`"evidence": ""`

`"SQL": "SELECT GasStationID FROM transactions_1k GROUP BY GasStationID ORDER BY SUM(Price) DESC LIMIT 1"`

`"difficulty": "simple"`

`"Predicted SQL": "SELECT T1.GasStationID FROM (SELECT GasStationID, SUM(Amount * Price) AS TotalRevenue FROM transactions_1k WHERE Amount IS NOT NULL AND Price IS NOT NULL GROUP BY GasStationID) AS T1 ORDER BY T1.TotalRevenue DESC LIMIT 1;"`

`analysis : Incorrect golden SQL`

`the revenue is calculated by the formula Amount * Price not price only (it is only the price of 1 transaction)`

▼ question 1533:

`"db_id": "debit_card_specializing"`

`"question": "For all the people who paid more than 29.00 per unit of product id No.5. Give their consumption status in the August of 2012."`

`"evidence": "August of 2012 means Date contains '201208' in the yearmonth.date of the database; Price per unit of product = Price / Amount"`

`"SQL": "SELECT T2.Consumption FROM transactions_1k AS T1 INNER JOIN yearmonth AS T2 ON T1.CustomerID = T2.CustomerID WHERE T1.Price / T1.Amount > 29.00 AND T1.ProductID = 5 AND T2.Date = '201208'"`

`"difficulty": "moderate"`

`"Predicted SQL": "SELECT T2.CustomerID, T2.Consumption FROM transactions_1k T1 JOIN yearmonth T2 ON T1.CustomerID = T2.CustomerID WHERE T1.ProductID = 5 AND T1.Price / T1.Amount > 29.00 AND T2.Date = '201208';"`

`analysis : Incorrect column output`

`the question doesn't specify exactly what columns to include in the result so the predicted sql added the customerID column which is more clear than the golden sql to know who are the people.`