Used approach: EditSQL (https://github.com/ryanzhumich/editsql)

Successfully working for the following SQL operations:

- SELECT
- ORDER BY (with LIMIT & DESC)
- GROUP BY
- COUNT
- MIN, MAX, AVERAGE
- JOIN
- BETWEEN
- WHERE (column name only)

Sample questions and predictions:

Good predictions:

```
Q: Which employee in the hospital gets paid the most ?
```

Prediction:

```
 select employee.* order_by employee.salary desc limit_value
```

```
Q: What is the maximum and minimum salary in the hospital ?
```

Prediction:

```
select max ( employee.salary ) , min ( employee.salary )
```

```
Q: Which patient has the maximum number of visitors ?
```

Prediction:

```
select patient.* group_by visitor.rid order_by count ( visitor.* ) desc
limit_value
```

```
Q: How many employees in the hospital are male and older than 65 ?
```

Prediction:

```
select count ( employee.* ) where people.sex = value and people.age >
value
```

```
Q: How many employees for each salary ?
```

Prediction:

```
select count ( employee.* ) , employee.salary group_by employee.salary
```

Some partially correct predictions:

```
Q: Which patient spent most on medicines ?

select patient.* group_by prescription.pid order_by count ( patient.* )
desc limit_value

Q: Which room accommodates the patient who has the maximum spending on
medication ?

select patient.* where medical_history.symptoms = value order_by
prescription.price desc limit_value _EOS

Q: What are the different types of nurses and doctors in the hospital

select distinct nurse.type
```

[Link to demo video]()

Query Distribution

SQL Clauses	Working Queries	Total Queries
Order By	6	10
Group By	4	5
Aggregate functions (max, min, count, sum)	15	17
Where	11	17

Total Queries: 35
Wrong Queries: 8

Queries with “WHERE” clauses also work successfully in case of other datasets like ATIS and WIKISQL. Some state of the art models(for WikiSQL) like SQLNet and SyntaxSQL make use of sequence to sequence models. They employ sequence-to-set prediction using column attention. The basic intuition behind this approach is that the column names appearing in the WHERE clause constitute a subset of the full set of all column names.

SQLNet first predicts the set of columns that appear in WHERE clause as explained above and then for each column it generates OP and VALUE slots. For each column in the WHERE clause , predicting the value of its OP slot is a 3 way classification: the model needs to choose from {=,>,<}. And for the VALUE slot, they predict a substring from the natural language question. To this end, SQLNet employs a sequence to sequence structure to generate the substring. Since order of the tokens in the VALUE slot matters, using a seq2seq structure is reasonable.

Thus,instead of generating a sequence of column names, prediction based on interest can be done. This approach of column attention is difficult to apply in the case of the EDITSQL model as it uses a contextual knowledge based architecture to handle the task of conversion from text to SQL query on SPIDER dataset.

As of now, this model cannot predict the value after the WHERE clause. For example, the question: `How many doctors have speciality in neurology?` gives the correct prediction:
`select count ( doctor.* ) where doctor.specialty = value`
except for the value after the WHERE clause where it should say `neurology` but for now a default placeholder `value` is placed instead.

Due to lack of research to solve this problem there are not many ways to accurately predict the value. However, there is a method that works. The AllenNLP team has created a system that successfully predicts the entire SQL query, including the values for WHERE.(https://demo.allennlp.org/atis-parser/MTM3MTcyNg==). For now it works only for the ATIS dataset and further work will have to be done to make it work for a custom dataset. We chose EditSQL since it works for custom datasets.