Implementing creative ways of generating SQL Queries in a Data Exploration Session

Krishna Prasad Sheshadri, Computer Science
Mentor: Professor Mohamed Sarwat Abdelghany Aly Elsayed; Collaborator: Vamsi Meduri
CIDSE, Arizona State University

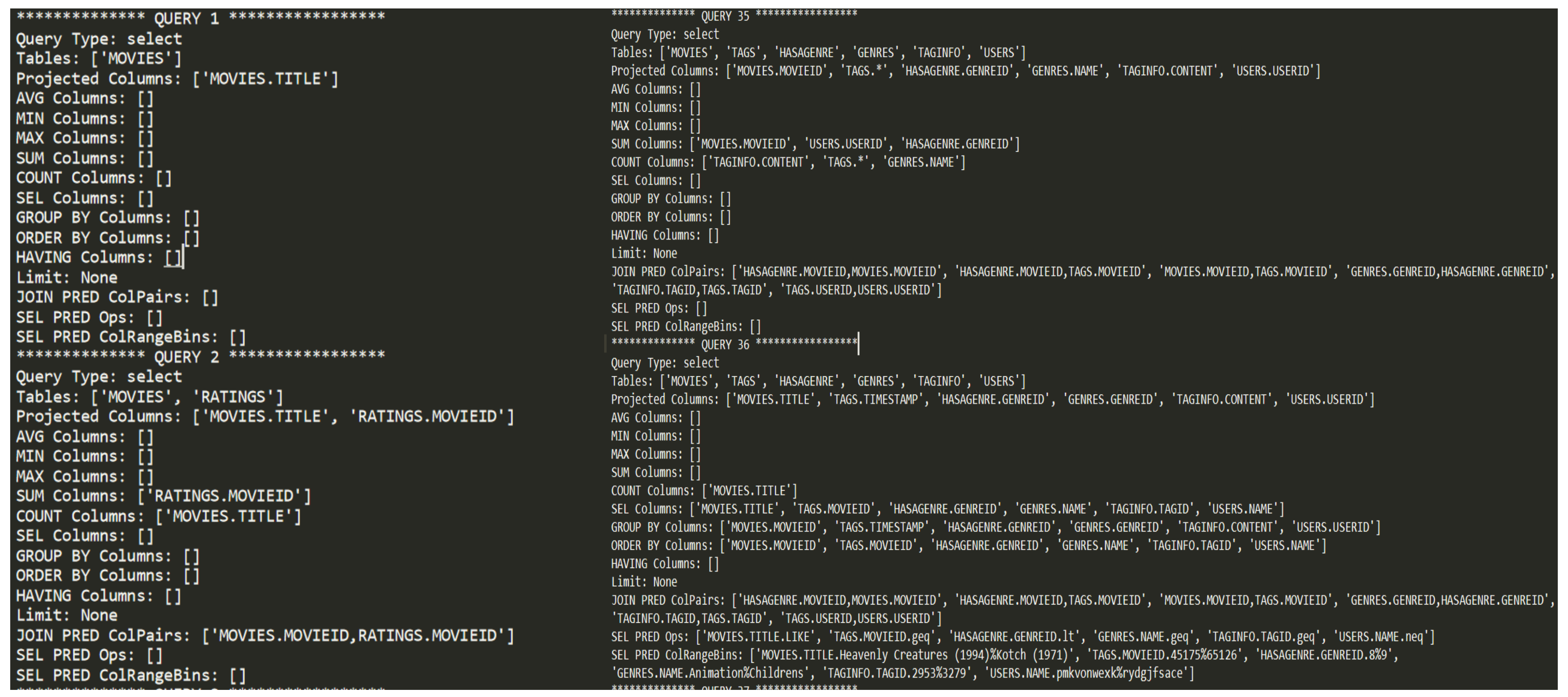
Research question: The objective of this research is to develop creative ways of generating SQL queries during a Data Exploration session. This research aims to develop efficient method of the collecting human interactions on databases. In order to achieve this, the research team has been trying and implementing various methods of Data collection. In this research, we explore three such methods and evaluate their benefits and shortcomings.

Introduction: An Interactive Data Exploration session primarily consists of a user interacting with a dataset to draw interesting insights/ patterns by issuing a sequence of queries. Database Researchers require large amounts of diverse user sessions, to train ML models and support predictive analytics. Unfortunately, having access to industrial database SQL logs is close to impossible as these are held proprietary. So, in order to serve the academic database community which may not have an insider access to large-scale OLAP workloads but are intending to perform research on predicting the human-intent during an exploration session, we propose and evaluate creative ways of efficiently generating such SQL queries.

Methods:

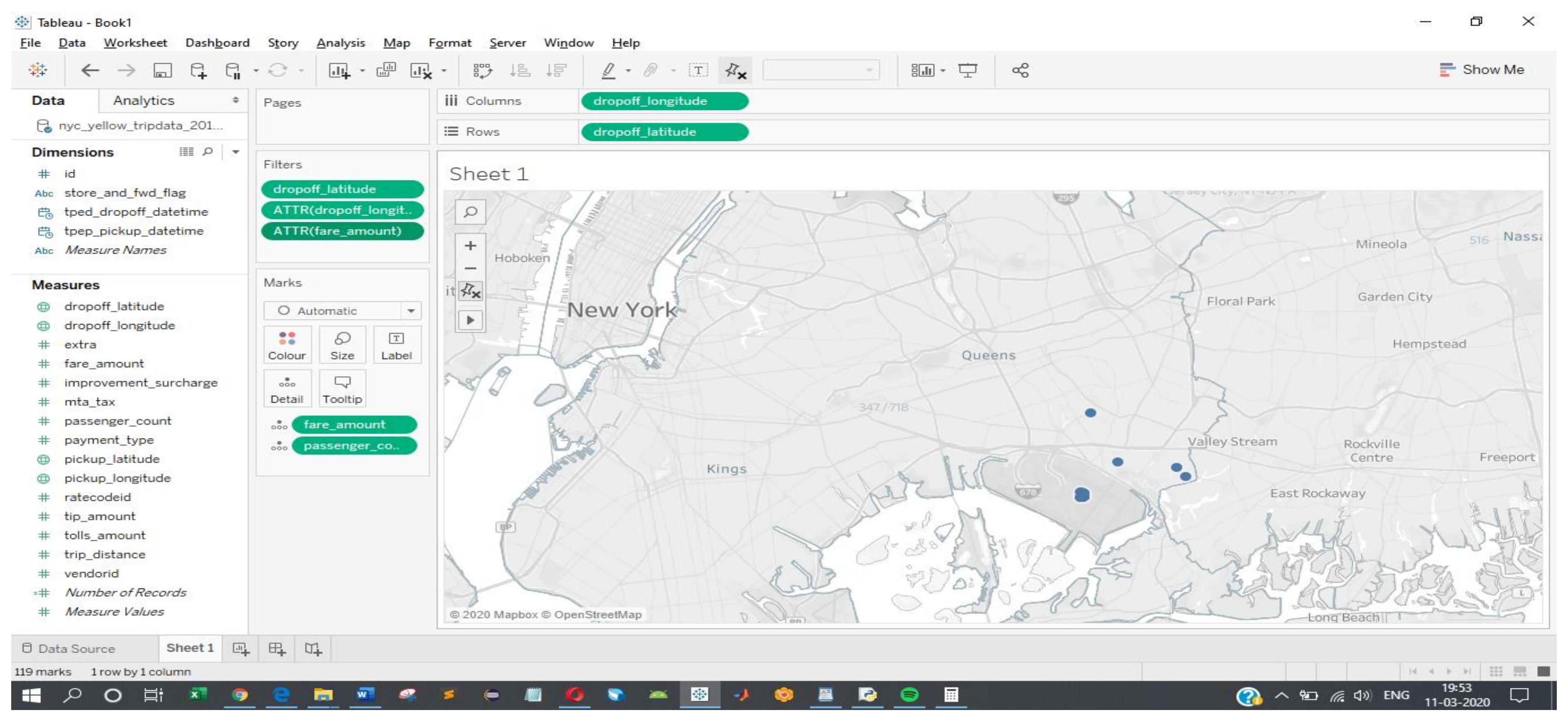
1. Using Tableau: The first method of generating sessions was through the Data Visualization tool, Tableau. I generated various sessions on three datasets: NYCTaxiTrips, Movies dataset and Yelp dataset. A typical interactive exploratory session on a dataset would start with a user issuing a simple query at first and the queries getting progressively more complex to explore interesting patterns and results in the dataset. The advantage of this method was that a user could interact with a dataset easily via the Tableau's interface. All the interactions were being logged, on which we would run a parser code to extract the SQL queries. However, the disadvantage was the lack of accessibility of this method to a wide number of users, as each user who wanted to interact with a dataset would have to download the application and the datasets on their local machines.

2. Synthetic Query Generator: The second approach was an auto-generator of SQL Queries. The query generator is probabilistically principled and uses a Bayesian model to synthesize user sessions. Bayes theorem can loosely be defined as Posterior ∝ Prior × Conditional. The prior probabilities simulate user behavior while transitioning from one query to another in a session and capture whether or not a specific SQL operator (WHERE, GROUP BY, JOIN..) exists in a SQL query. The conditional probabilities are modeled by well known statistical distributions such as Gaussian, Poisson, Uniform etc. and represent the likelihood with which a schema element (Column name, Table name..) can be associated with a chosen SQL operator. The limitation of this method is that the conditional probabilities drawn from statistical distributions are not guaranteed to reflect human behavior.

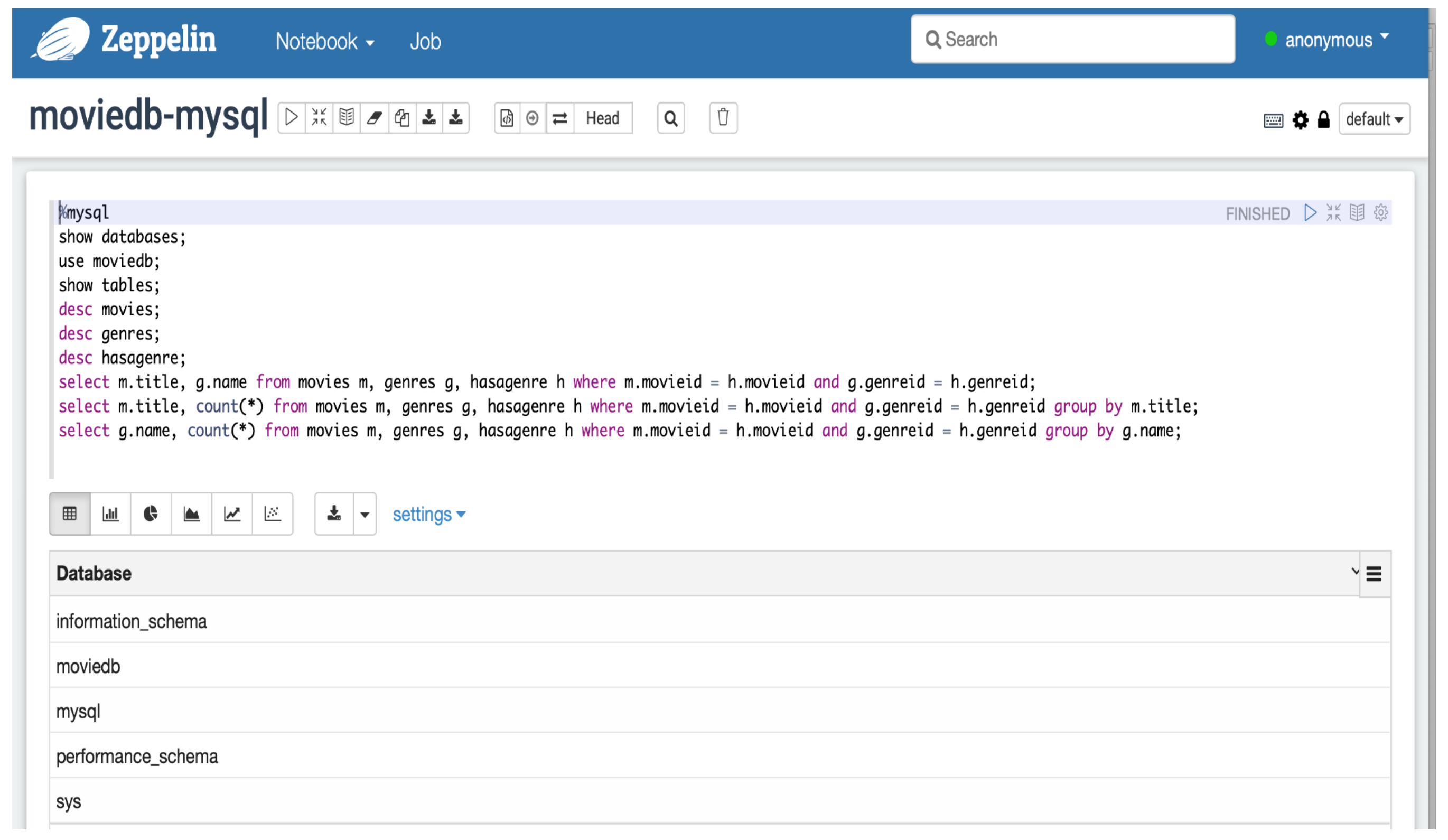


```
************** QUERY 1 *****************
Query Type: select
Tables: ['MOVIES']
Projected Columns: ['MOVIES.TITLE']
AVG Columns: []
MIN Columns: []
MAX Columns: []
SUM Columns: []
COUNT Columns: []
SEL Columns: []
GROUP BY Columns: []
ORDER BY Columns: []
HAVING Columns: []
Limit: None
JOIN PRED ColPairs: []
SEL PRED Ops: []
SEL PRED ColRangeBins: []
************** QUERY 2 *****************
Query Type: select
Tables: ['MOVIES', 'RATINGS']
Projected Columns: ['MOVIES.TITLE', 'RATINGS.MOVIEID']
AVG Columns: []
MIN Columns: []
MAX Columns: []
SUM Columns: ['RATINGS.MOVIEID']
COUNT Columns: ['MOVIES.TITLE']
SEL Columns: []
GROUP BY Columns: []
ORDER BY Columns: []
HAVING Columns: []
Limit: None
JOIN PRED ColPairs: ['MOVIES.MOVIEID,RATINGS.MOVIEID']
SEL PRED Ops: []
SEL PRED ColRangeBins: []
```

```
************** QUERY 35 *****************
Query Type: select
Tables: ['MOVIES', 'TAGS', 'HASAGENRE', 'GENRES', 'TAGINFO', 'USERS']
Projected Columns: ['MOVIES.MOVIEID', 'TAGS.*', 'HASAGENRE.GENREID', 'GENRES.NAME', 'TAGINFO.CONTENT', 'USERS.USERID']
AVG Columns: []
MIN Columns: []
MAX Columns: []
SUM Columns: ['MOVIES.MOVIEID', 'USERS.USERID', 'HASAGENRE.GENREID']
COUNT Columns: ['TAGINFO.CONTENT', 'TAGS.*', 'GENRES.NAME']
SEL Columns: []
GROUP BY Columns: []
ORDER BY Columns: []
HAVING Columns: []
Limit: None
JOIN PRED ColPairs: ['HASAGENRE.MOVIEID,MOVIES.MOVIEID', 'HASAGENRE.MOVIEID,TAGS.MOVIEID', 'MOVIES.MOVIEID,TAGS.MOVIEID', 'GENRES.GENREID,HASAGENRE.GENREID',
'TAGINFO.TAGID,TAGS.TAGID', 'TAGS.USERID,USERS.USERID']
SEL PRED Ops: []
SEL PRED ColRangeBins: []
************** QUERY 36 *****************
Query Type: select
Tables: ['MOVIES', 'TAGS', 'HASAGENRE', 'GENRES', 'TAGINFO', 'USERS']
Projected Columns: ['MOVIES.TITLE', 'TAGS.TIMESTAMP', 'HASAGENRE.GENREID', 'GENRES.GENREID', 'TAGINFO.CONTENT', 'USERS.USERID']
AVG Columns: []
MIN Columns: []
MAX Columns: []
SUM Columns: []
COUNT Columns: ['MOVIES.TITLE']
SEL Columns: ['MOVIES.TITLE', 'TAGS.MOVIEID', 'HASAGENRE.GENREID', 'GENRES.NAME', 'TAGINFO.TAGID', 'USERS.NAME']
GROUP BY Columns: ['MOVIES.MOVIEID', 'TAGS.TIMESTAMP', 'HASAGENRE.GENREID', 'GENRES.GENREID', 'TAGINFO.CONTENT', 'USERS.USERID']
ORDER BY Columns: ['MOVIES.MOVIEID', 'TAGS.MOVIEID', 'HASAGENRE.GENREID', 'GENRES.NAME', 'TAGINFO.TAGID', 'USERS.NAME']
HAVING Columns: []
Limit: None
JOIN PRED ColPairs: ['HASAGENRE.MOVIEID,MOVIES.MOVIEID', 'HASAGENRE.MOVIEID,TAGS.MOVIEID', 'MOVIES.MOVIEID,TAGS.MOVIEID', 'GENRES.GENREID,HASAGENRE.GENREID',
'TAGINFO.TAGID,TAGS.TAGID', 'TAGS.USERID,USERS.USERID']
SEL PRED Ops: ['MOVIES.TITLE.LIKE', 'TAGS.MOVIEID.geq', 'HASAGENRE.GENREID.lt', 'GENRES.NAME.geq', 'TAGINFO.TAGID.geq', 'USERS.NAME.neq']
SEL PRED ColRangeBins: ['MOVIES.TITLE.Heavenly Creatures (1994)%Kotch (1971)', 'TAGS.MOVIEID.45175%65126', 'HASAGENRE.GENREID.8%9',
'GENRES.NAME.Animation%Childrens', 'TAGINFO.TAGID.2953%3279', 'USERS.NAME.pmkvonwexk%rydgjfsace']
```

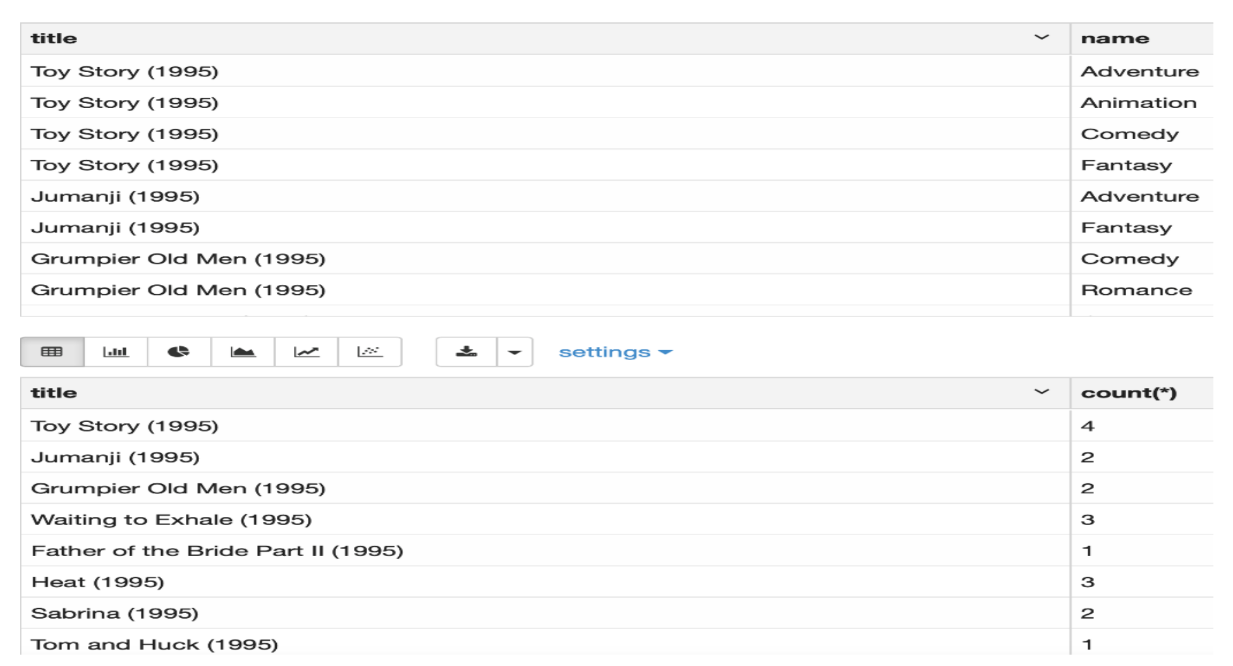


3.Zeppelin notebook: The most recent method we have implemented is an Apache Zeppelin notebook that is hosted on a server for multiple users to interact with the database at the same time. The users can issue SQL queries on relational databases to perform interactive data exploration. To interact with geospatial datasets users use Spark SQL.



```
%mysql
show databases;
use moviedb;
show tables;
desc movies;
desc genres;
desc hasagenre;
select m.title, g.name from movies m, genres g, hasagenre h where m.movieid = h.movieid and g.genreid = h.genreid;
select m.title, count(*) from movies m, genres g, hasagenre h where m.movieid = h.movieid and g.genreid = h.genreid group by m.title;
select g.name, count(*) from movies m, genres g, hasagenre h where m.movieid = h.movieid and g.genreid = h.genreid group by g.name;
```

Database
information_schema
moviedb
mysql
performance_schema
sys



title	name
Toy Story (1995)	Adventure
Toy Story (1995)	Animation
Toy Story (1995)	Comedy
Toy Story (1995)	Fantasy
Jumanji (1995)	Adventure
Jumanji (1995)	Fantasy
Grumpier Old Men (1995)	Comedy
Grumpier Old Men (1995)	Romance

title	count(*)
Toy Story (1995)	4
Jumanji (1995)	2
Grumpier Old Men (1995)	2
Waiting to Exhale (1995)	3
Father of the Bride Part II (1995)	1
Heat (1995)	3
Sabrina (1995)	2
Tom and Huck (1995)	1

Future Studies: All the above-mentioned methods to generate SQL queries in an Interactive Data Exploration (IDE) session have their limitations. Going forward, we plan to design a synthetic query generator which shall learn the conditional probabilities from the human sessions using a Machine Learning model. However, we shall also explore novel and efficient ways to collect more interactive sessions from real users.

References:
1. Venkata Vamsikrishna Meduri, Kanchan Chowdhury, Mohamed Sarwat: Recurrent Neural Networks for Dynamic User Intent Prediction in Human-Database Interaction. EDBT 2019: 654-657
2. *Magdalini Eirinaki*, Suju Abraham, Neoklis Polyzotis, Naushin Shaikh: ***QueRIE: Collaborative Database Exploration.*** *IEEE Trans. Knowl. Data Eng. 26(7)*: 1778-1790 (2014)