# Workload Design

Ankit Kangale
2019CS10363

Rayyan Shahid
2019CS10392

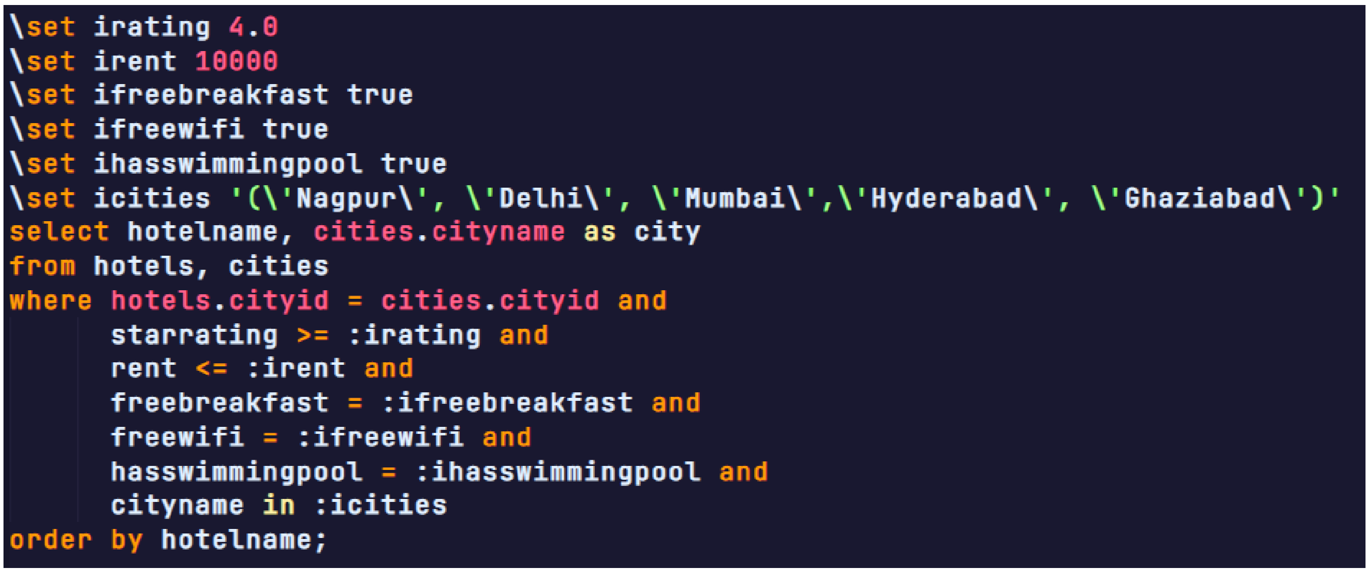
Parth Gupta
2019CS10380

March 2023

## 1 Queries

We have implemented the following queries:

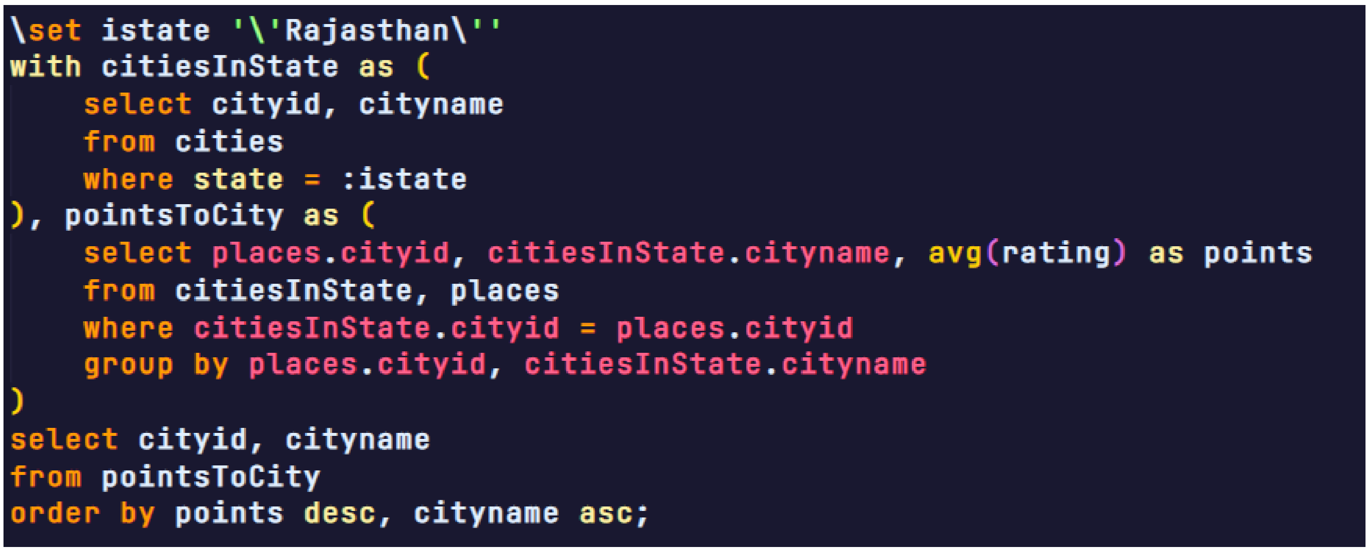
1. Filter on Hotels: Display hotels with specified filters (eg. rent, rating, list of cities, other facilities etc.)



```
\set irating 4.0
\set irent 10000
\set ifreebreakfast true
\set ifreewifi true
\set ihasswimmingpool true
\set icities '(\'Nagpur\', \'Delhi\', \'Mumbai\',\'Hyderabad\', \'Ghaziabad\')'
select hotelname, cities.cityname as city
from hotels, cities
where hotels.cityid = cities.cityid and
      starrating >= :irating and
      rent <= :irent and
      freebreakfast = :ifreebreakfast and
      freewifi = :ifreewifi and
      hasswimmingpool = :ihasswimmingpool and
      cityname in :icities
order by hotelname;
```

Figure 1: Query 1

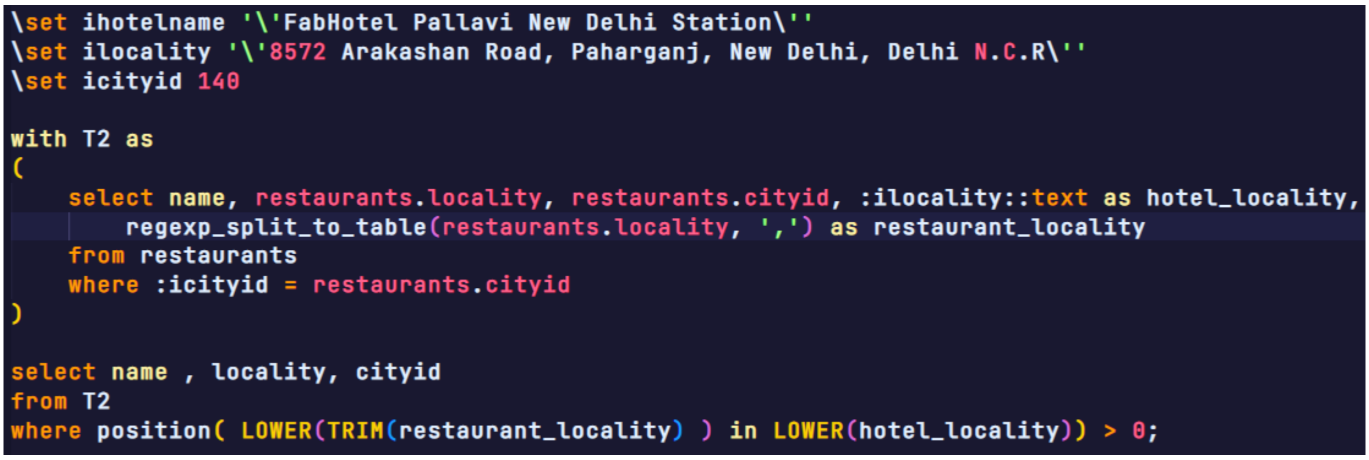
2. Top cities in a State: Given a state, display the cities in the state with the highest average of the ratings of the places in that city.



```
\set istate '\'Rajasthan\''
with citiesInState as (
    select cityid, cityname
    from cities
    where state = :istate
), pointsToCity as (
    select places.cityid, citiesInState.cityname, avg(rating) as points
    from citiesInState, places
    where citiesInState.cityid = places.cityid
    group by places.cityid, citiesInState.cityname
)
select cityid, cityname
from pointsToCity
order by points desc, cityname asc;
```

Figure 2: Query 2

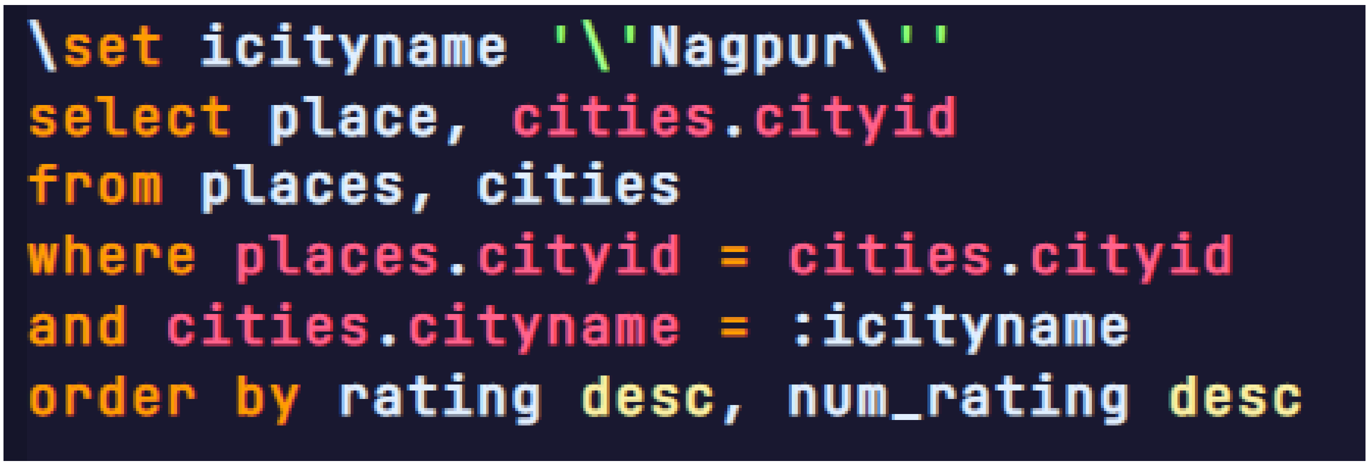
3. Restaurants near Hotel: Given a hotel, display the restaurants in the same locality as the hotel.



```
\set ihotelname '\'FabHotel Pallavi New Delhi Station\''
\set ilocality '\'8572 Arakashan Road, Paharganj, New Delhi, Delhi N.C.R\''
\set icityid 140

with T2 as
(
    select name, restaurants.locality, restaurants.cityid, :ilocality::text as hotel_locality,
        regexp_split_to_table(restaurants.locality, ',') as restaurant_locality
    from restaurants
    where :icityid = restaurants.cityid
)

select name , locality, cityid
from T2
where position( LOWER(TRIM(restaurant_locality) ) in LOWER(hotel_locality)) > 0;
```

Figure 3: Query 3

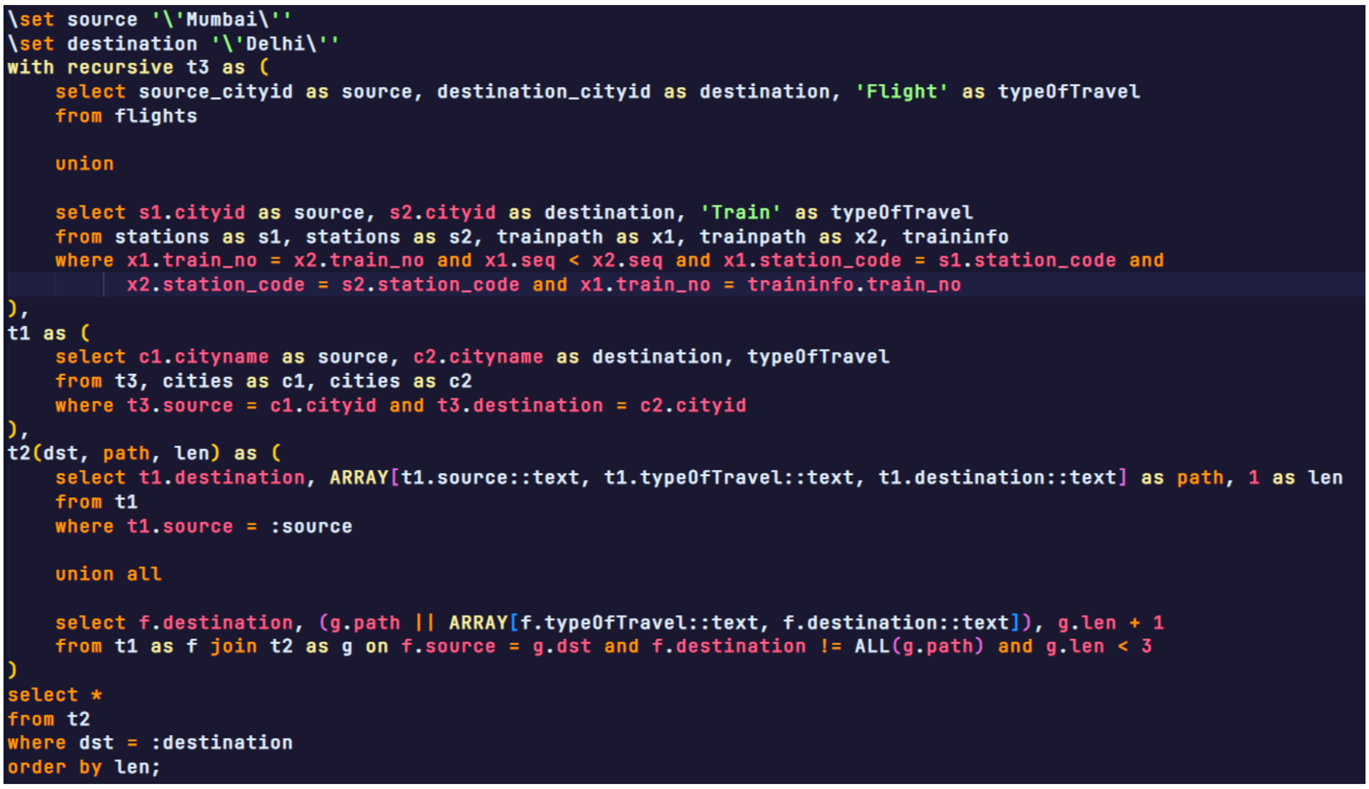
4. Places in City: Given a city, display the places to visit in that city sorted in descending order by their rating and number of ratings.



```
\set icityname '\'Nagpur\''
select place, cities.cityid
from places, cities
where places.cityid = cities.cityid
and cities.cityname = :icityname
order by rating desc, num_rating desc
```

Figure 4: Query 4

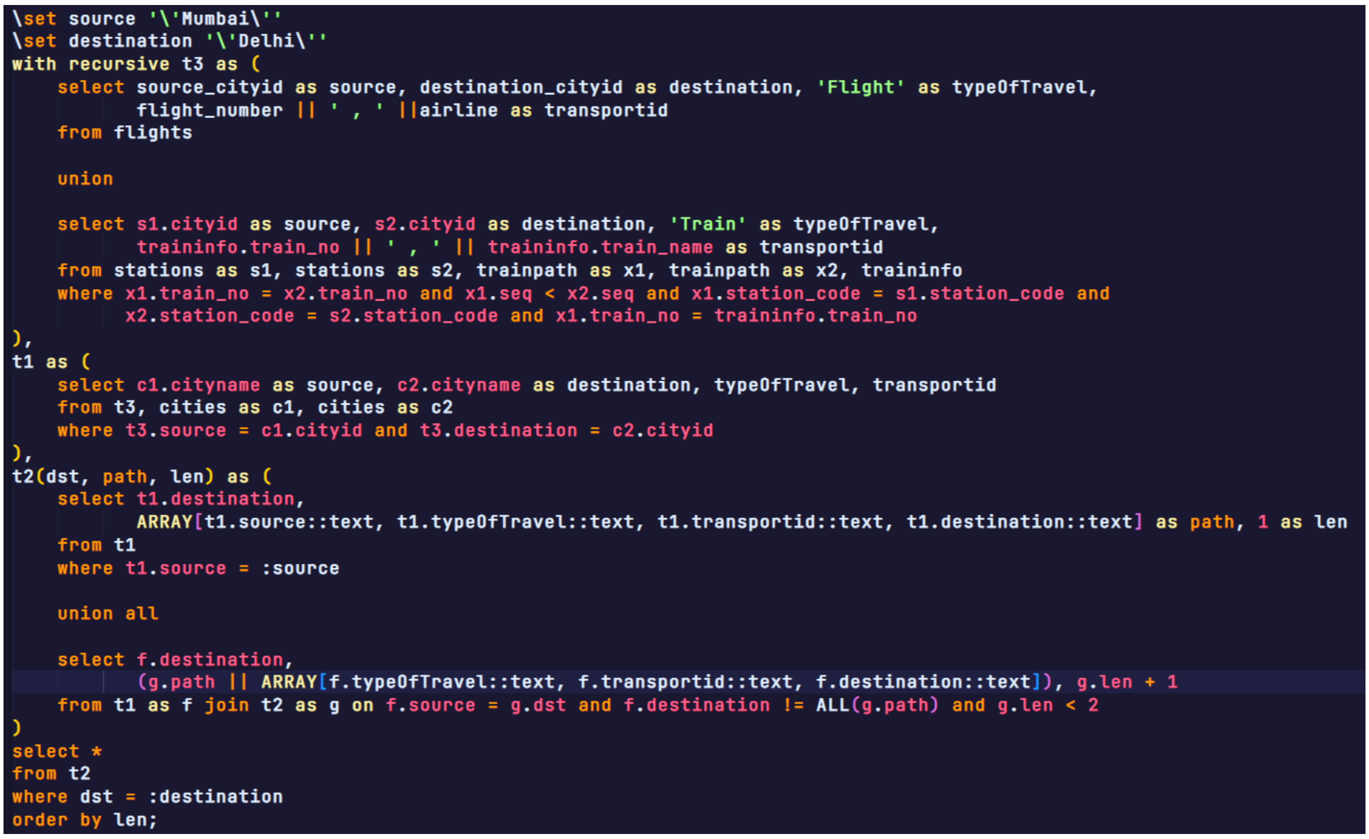
5. Path between Cities: Given a source city and a destination city, find all the paths (via flights and trains) from source to destination using atmost 3 transports.(eg. (flight, train); (train, flight, flight) etc.)



```
\set source '\'Mumbai\''
\set destination '\'Delhi\''
with recursive t3 as (
    select source_cityid as source, destination_cityid as destination, 'Flight' as typeOfTravel
    from flights

    union

    select s1.cityid as source, s2.cityid as destination, 'Train' as typeOfTravel
    from stations as s1, stations as s2, trainpath as x1, trainpath as x2, traininfo
    where x1.train_no = x2.train_no and x1.seq < x2.seq and x1.station_code = s1.station_code and
          x2.station_code = s2.station_code and x1.train_no = traininfo.train_no
),
t1 as (
    select c1.cityname as source, c2.cityname as destination, typeOfTravel
    from t3, cities as c1, cities as c2
    where t3.source = c1.cityid and t3.destination = c2.cityid
),
t2(dst, path, len) as (
    select t1.destination, ARRAY[t1.source::text, t1.typeOfTravel::text, t1.destination::text] as path, 1 as len
    from t1
    where t1.source = :source

    union all

    select f.destination, (g.path || ARRAY[f.typeOfTravel::text, f.destination::text]), g.len + 1
    from t1 as f join t2 as g on f.source = g.dst and f.destination != ALL(g.path) and g.len < 3
)
select *
from t2
where dst = :destination
order by len;
```

Figure 5: Query 5

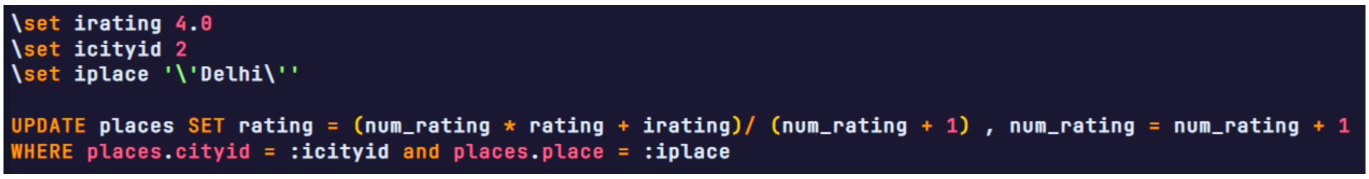
6. Path between Cities With Mode of Transport: Given a source city and a destination city, find all the paths and the mode of transport used (via flights and trains) from source to destination using atmost 2 transports.(eg. (flight, train); (train, flight, flight) etc.)



```
\set source '\'Mumbai\''
\set destination '\'Delhi\''
with recursive t3 as (
    select source_cityid as source, destination_cityid as destination, 'Flight' as typeOfTravel,
           flight_number || ' , ' ||airline as transportid
    from flights

    union

    select s1.cityid as source, s2.cityid as destination, 'Train' as typeOfTravel,
           traininfo.train_no || ' , ' || traininfo.train_name as transportid
    from stations as s1, stations as s2, trainpath as x1, trainpath as x2, traininfo
    where x1.train_no = x2.train_no and x1.seq < x2.seq and x1.station_code = s1.station_code and
          x2.station_code = s2.station_code and x1.train_no = traininfo.train_no
),
t1 as (
    select c1.cityname as source, c2.cityname as destination, typeOfTravel, transportid
    from t3, cities as c1, cities as c2
    where t3.source = c1.cityid and t3.destination = c2.cityid
),
t2(dst, path, len) as (
    select t1.destination,
           ARRAY[t1.source::text, t1.typeOfTravel::text, t1.transportid::text, t1.destination::text] as path, 1 as len
    from t1
    where t1.source = :source

    union all

    select f.destination,
           (g.path || ARRAY[f.typeOfTravel::text, f.transportid::text, f.destination::text]), g.len + 1
    from t1 as f join t2 as g on f.source = g.dst and f.destination != ALL(g.path) and g.len < 2
)
select *
from t2
where dst = :destination
order by len;
```

Figure 6: Query 6

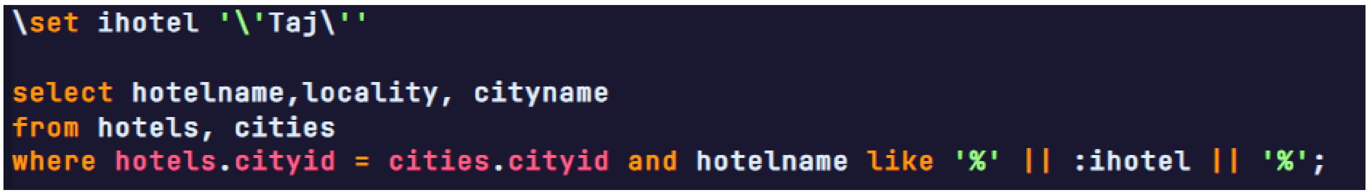
7. Update Place Rating: Update the rating and num_rating of places when a user rates a place.



```
\set irating 4.0
\set icityid 2
\set iplace '\'Delhi\''

UPDATE places SET rating = (num_rating * rating + irating)/ (num_rating + 1) , num_rating = num_rating + 1
WHERE places.cityid = :icityid and places.place = :iplace
```

Figure 7: Query 7

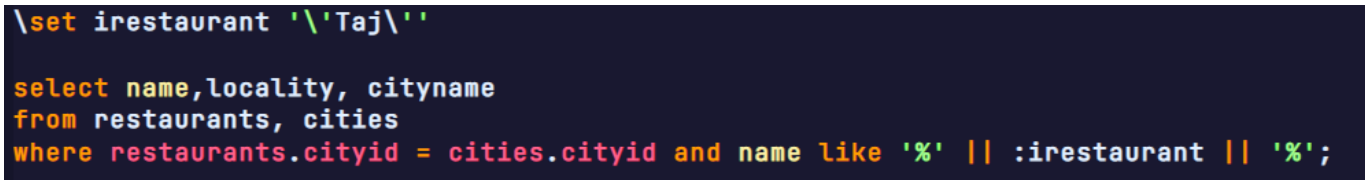
8. Hotel Search: Find all hotels (city, locality) with the given name.



```
\set ihotel '\'Taj\''

select hotelname,locality, cityname
from hotels, cities
where hotels.cityid = cities.cityid and hotelname like '%' || :ihotel || '%';
```

Figure 8: Query 8

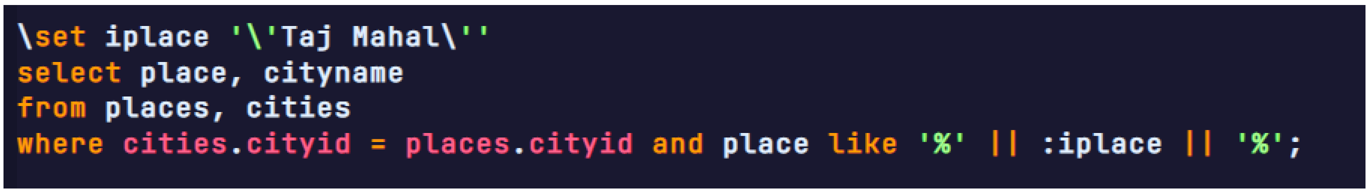
9. Restaurant Search: Find all restaurants (city, locality) with the given name.



```
\set irestaurant '\'Taj\''

select name,locality, cityname
from restaurants, cities
where restaurants.cityid = cities.cityid and name like '%' || :irestaurant || '%';
```

Figure 9: Query 9

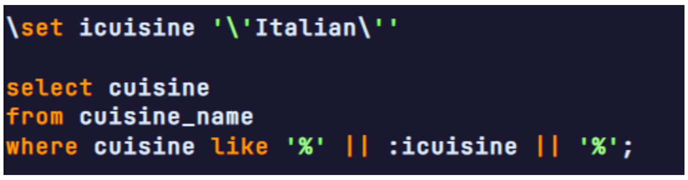
10. Place Search: Find the city which has the given place.



```
\set iplace '\'Taj Mahal\''
select place, cityname
from places, cities
where cities.cityid = places.cityid and place like '%' || :iplace || '%';
```

Figure 10: Query 10

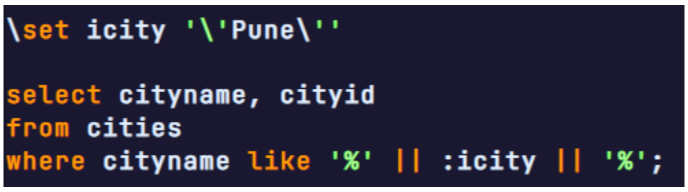
11. search Cuisine names:



```
\set icuisine '\'Italian\''

select cuisine
from cuisine_name
where cuisine like '%' || :icuisine || '%';
```

Figure 11: Query 11

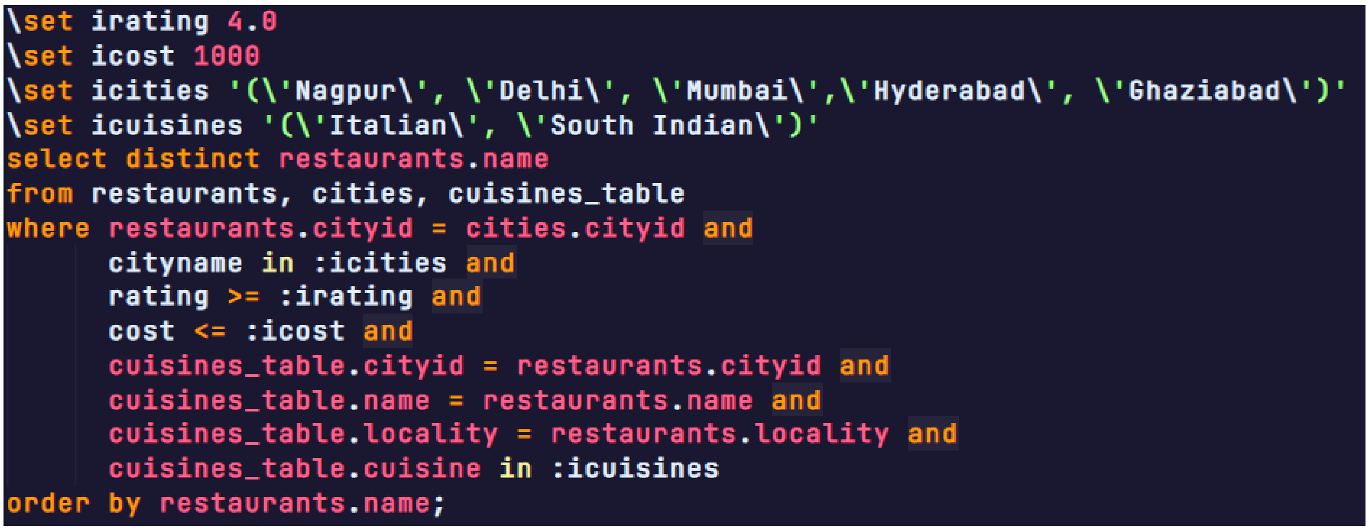
12. search City names:



```
\set icity '\'Pune\''

select cityname, cityid
from cities
where cityname like '%' || :icity || '%';
```

Figure 12: Query 12

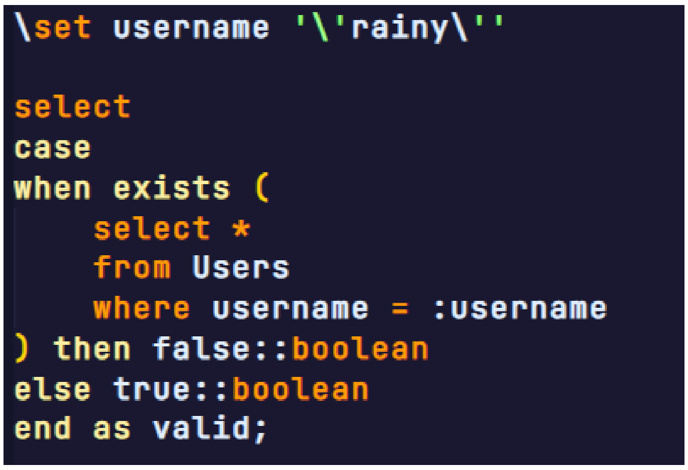
13. Filter on Restaurants: Display restaurants with specified filters (eg. cost, rating, list of cities, list of cuisines)



```
\set irating 4.0
\set icost 1000
\set icities '(\'Nagpur\', \'Delhi\', \'Mumbai\',\'Hyderabad\', \'Ghaziabad\')'
\set icuisines '(\'Italian\', \'South Indian\')'
select distinct restaurants.name
from restaurants, cities, cuisines_table
where restaurants.cityid = cities.cityid and
      cityname in :icities and
      rating >= :irating and
      cost <= :icost and
      cuisines_table.cityid = restaurants.cityid and
      cuisines_table.name = restaurants.name and
      cuisines_table.locality = restaurants.locality and
      cuisines_table.cuisine in :icuisines
order by restaurants.name;
```

Figure 13: Query 13

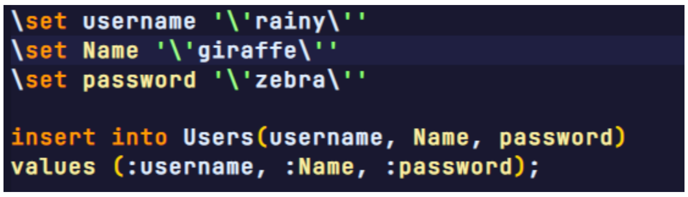
14. Validate user registration: Returns true if no user with the given username exists.



```
\set username '\'rainy\''

select
case
when exists (
    select *
    from Users
    where username = :username
) then false::boolean
else true::boolean
end as valid;
```

Figure 14: Query 14

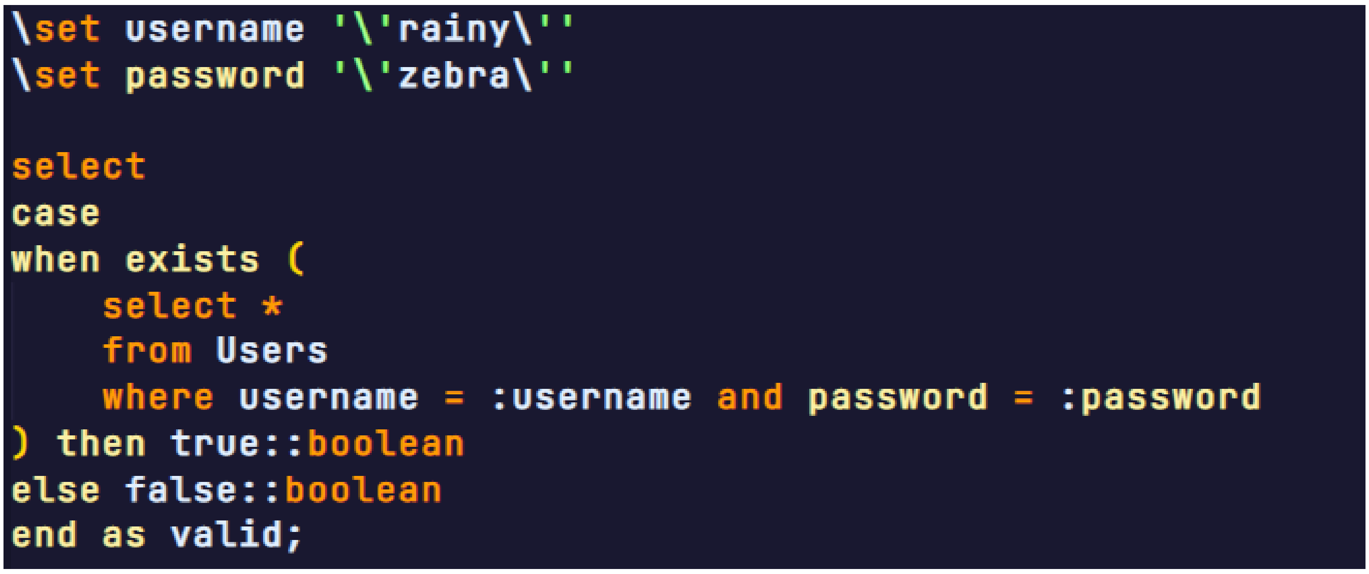
15. Add user details: Add username, name and password hash to the Users table



```
\set username '\'rainy\''
\set Name '\'giraffe\''
\set password '\'zebra\''

insert into Users(username, Name, password)
values (:username, :Name, :password);
```

Figure 15: Query 15

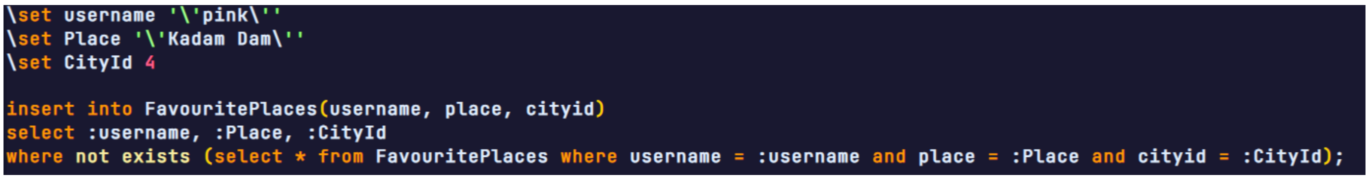
16. Authenticate user login: Authenticate login using username and password hash.



```
\set username '\'rainy\''
\set password '\'zebra\''

select
case
when exists (
    select *
    from Users
    where username = :username and password = :password
) then true::boolean
else false::boolean
end as valid;
```

Figure 16: Query 16

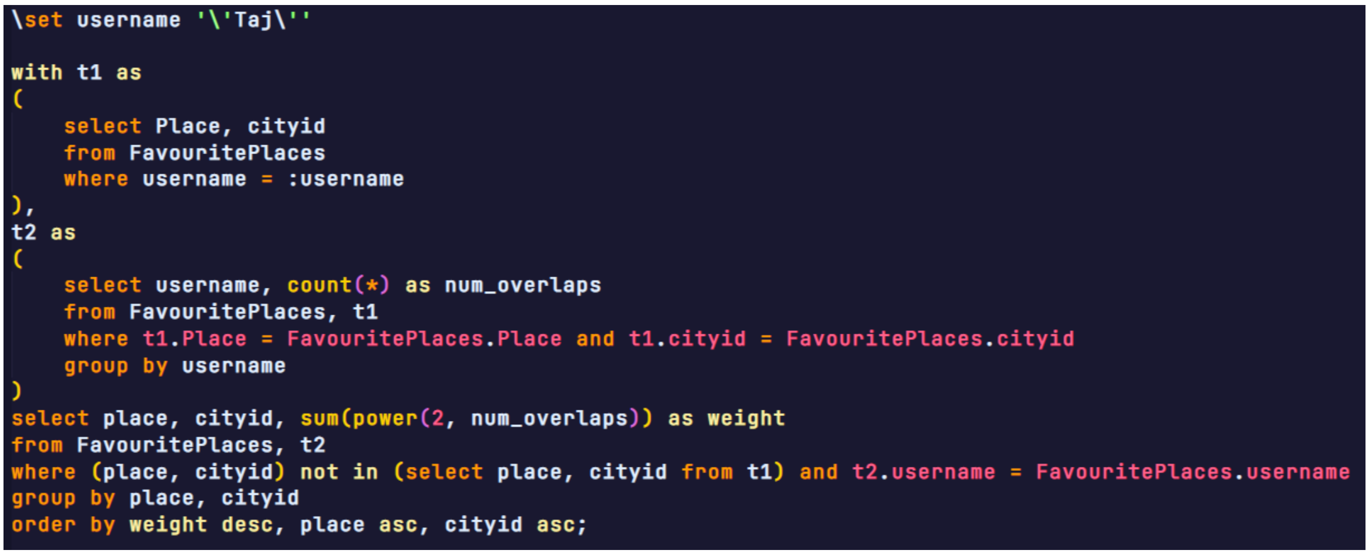
17. Insert to favourite places: Add a place to favourite places table for the specified user



```
\set username '\'pink\''
\set Place '\'Kadam Dam\''
\set CityId 4

insert into FavouritePlaces(username, place, cityid)
select :username, :Place, :CityId
where not exists (select * from FavouritePlaces where username = :username and place = :Place and cityid = :CityId);
```

Figure 17: Query 17

18. Recommend places to visit: Given a user and its favourite places, recommend places to visit based on searches of other users who prefer similar places.



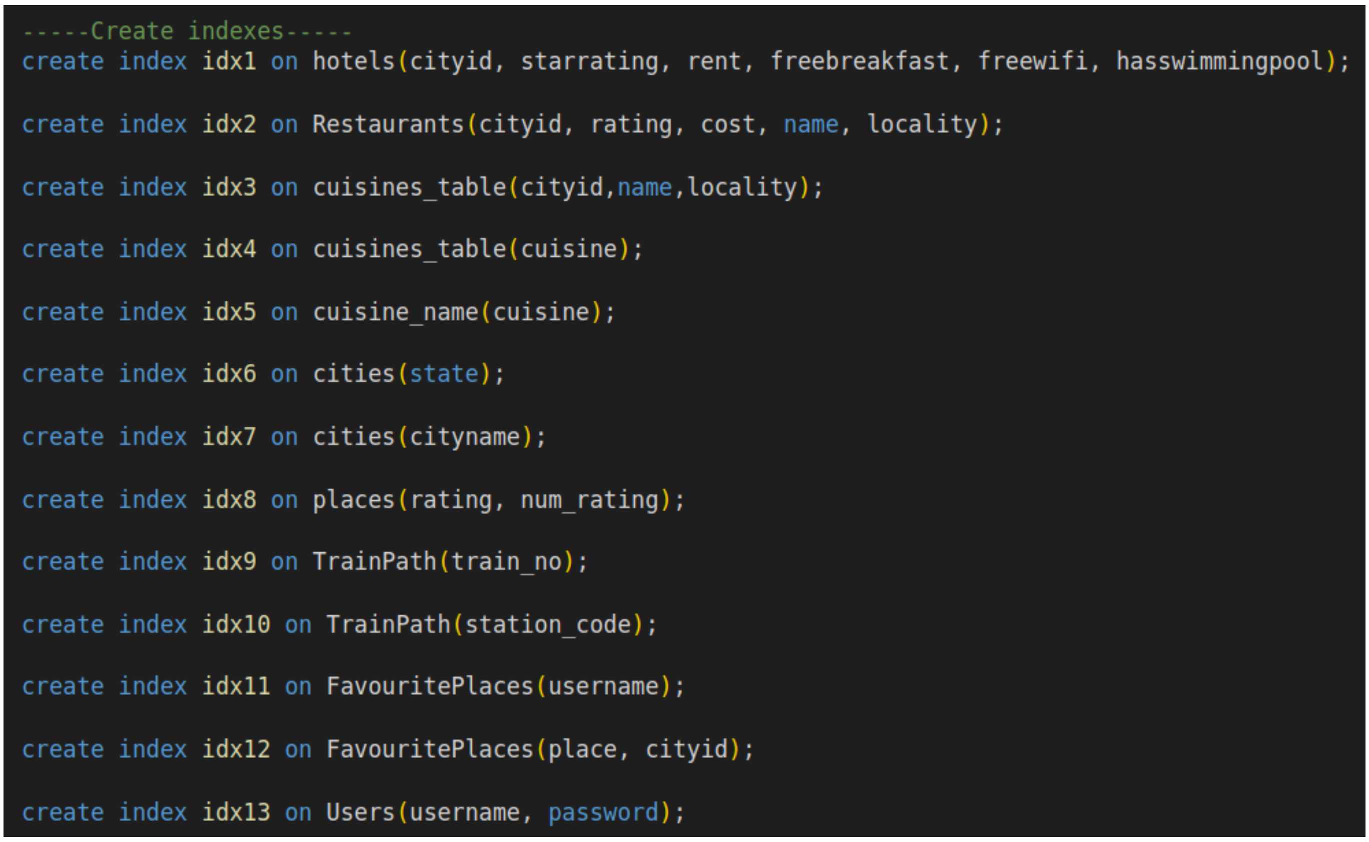
```
\set username '\'Taj\''

with t1 as
(
    select Place, cityid
    from FavouritePlaces
    where username = :username
),
t2 as
(
    select username, count(*) as num_overlaps
    from FavouritePlaces, t1
    where t1.Place = FavouritePlaces.Place and t1.cityid = FavouritePlaces.cityid
    group by username
)
select place, cityid, sum(power(2, num_overlaps)) as weight
from FavouritePlaces, t2
where (place, cityid) not in (select place, cityid from t1) and t2.username = FavouritePlaces.username
group by place, cityid
order by weight desc, place asc, cityid asc;
```

Figure 18: Query 18

## 2 Optimizations

### 2.1 Index Choices

The following index choices were made based on the above queries.

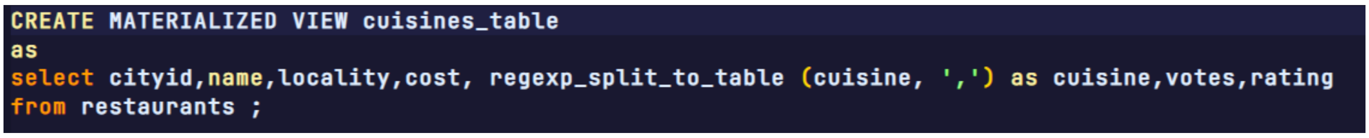


```
-----Create indexes-----
create index idx1 on hotels(cityid, starrating, rent, freebreakfast, freewifi, hasswimmingpool);

create index idx2 on Restaurants(cityid, rating, cost, name, locality);

create index idx3 on cuisines_table(cityid,name,locality);

create index idx4 on cuisines_table(cuisine);

create index idx5 on cuisine_name(cuisine);

create index idx6 on cities(state);

create index idx7 on cities(cityname);

create index idx8 on places(rating, num_rating);

create index idx9 on TrainPath(train_no);

create index idx10 on TrainPath(station_code);

create index idx11 on FavouritePlaces(username);

create index idx12 on FavouritePlaces(place, cityid);

create index idx13 on Users(username, password);
```

Figure 19: Indexes

## 2.2 Materialized views

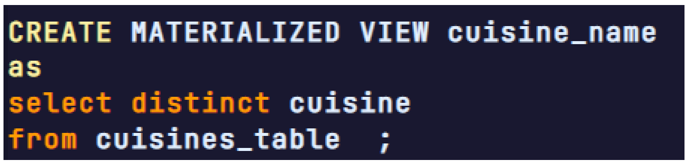
1. View for Cuisines: Create a materialized view for cuisines in each restaurant.



```
CREATE MATERIALIZED VIEW cuisines_table
as
select cityid,name,locality,cost, regexp_split_to_table (cuisine, ',') as cuisine,votes,rating
from restaurants ;
```

Figure 20: View 1

2. View for storing Cuisine names: Create a materialized view for storing all distinct cuisines



```
CREATE MATERIALIZED VIEW cuisine_name
as
select distinct cuisine
from cuisines_table  ;
```

Figure 21: View 2

## 3 Database Size and Performance

Database performance on allocated server

| Query | Time |
|---|---|
| Q1 | 10.702 ms |
| Q2 | 16.675 ms |
| Q3 | 29.409 ms |
| Q4 | 8.137 ms |
| Q5 | 1290.097 ms |
| Q6 | 3070.309 ms |
| Q7 | 2.116 ms |
| Q8 | 4.352 ms |
| Q9 | 4.603 ms |
| Q10 | 3.616 ms |
| Q11 | 1.101 ms |
| Q12 | 1.623 ms |
| Q13 | 19.790 ms |

Database size

| Table | Number of tuples |
|---|---|
| cities | 1215 |
| favouriteplaces | 0 |
| flights | 4827 |
| hotels | 1393 |
| places | 5330 |
| restaurants | 6434 |
| stations | 494 |
| traininfo | 2569 |
| trainpath | 12804 |
| users | 0 |

Data dump size - 1.5 MB