|  |  |  |
| --- | --- | --- |
| Function | SQL | Mongo |
| Login | SELECT \* FROM user WHERE email = ? | loginModel.find({         email: lemail  }); |
| Register | INSERT INTO user SET ? | var registerData = new registerModel({         email: remail,         username: rusername,         password: rpassword,         created\_time: rtimer     });  registerData.save(); |
| Housing | SELECT  Distinct longitude, latitude,  housing.id, postal\_code, block, year, floors, GROUP\_CONCAT(room.room\_type)  As Rooms  FROM housing  JOIN room  ON housing.room\_id = room.id  WHERE  Locale\_name  LIKE  " + "'%" + locale + "%'" + "  AND year >  "+ syear +"  AND year <  "+ eyear +"  AND room\_id in "+room+"  GROUP BY longitude, latitude; | housingModel.find({         locale: locale         year: { '$gt': syear, '$lt': eyear },         rooms : roomType  }); |
| Primary | SELECT sh.\*,  (6371 \* acos(cos( radians (@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM school\_distance sh  WHERE sh.level = 'Primary'  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | primaryModel.find({         level: "PRIMARY",         location: {          $near: {             $maxDistance: (1000 \* proximity),             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| Secondary | SELECT sh.\*,  (6371 \* acos(cos( radians (@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM school\_distance sh  WHERE sh.level = ‘Secondary’'  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | secondaryModel.find({         level: "SECONDARY",         location: {          $near: {             $maxDistance: 1000 \* proximity,             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| Combined | SELECT sh.\*,  (6371 \* acos(cos( radians (@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM school\_distance sh  WHERE sh.level <> ‘Secondary’  AND sh.level <> ‘Primary’'  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | combinedModel.find({         level: {$nin : ["PRIMARY", "SECONDARY"]},         location: {          $near: {             $maxDistance: 1000 \* proximity,             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| Hawker | SELECT h.\*,  (6371 \* acos(cos( radians (@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM hawker\_distance h  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | hawkerModel.find({         location: {          $near: {             $maxDistance: 1000 \* proximity,             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| Bus-stop | SELECT bs.\*,  (6371 \* acos(cos( radians (@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM bus\_stop\_distance bs  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | bus\_stopModel.find({         location: {          $near: {             $maxDistance: 1000 \* proximity,             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| NPP | SELECT npc.\*,  (6371 \* acos(cos(radians(@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM npc\_distance npc  WHERE npc.npc\_name LIKE ‘%post%’  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | nppModel.find({         npc\_name: {$regex: /Police Post/},         location: {          $near: {             $maxDistance: 1000 \* proximity,             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| NPC | SELECT npc.\*,  (6371 \* acos(cos(radians(@latitude))  \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@longitude)) + sin(radians(@latitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM npc\_distance npc  WHERE npc.npc\_name NOT LIKE ‘%post%’  HAVING `distance` < @radiuskm  ORDER BY `distance` ASC; | npcModel.find({         npc\_name: {$regex: /Police Centre/},         location: {          $near: {             $maxDistance: 1000 \* proximity,             $geometry: {                 type: "Point",                 coordinates: [longitude, latitude]             }         }     } |
| Grant | SELECT  count(DISTINCT postal\_code) as eligible, (6371 \* acos(cos( radians (@queryLatitude)) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians(@queryLongitude)) + sin(radians(@queryLatitude)) \* sin(radians(`latitude`))))  AS `distance`  FROM housing  WHERE postal\_code = @user\_postal  HAVING `distance` < 4  ORDER BY `distance` ASC; | grantModel.find({         postal\_code: parentPostal     }, function(err,result){         if (err) throw err;         var parentCoordinates = result[0]['location']['coordinates'];  //Find coordinates for user based on parent's coordinates of 4km         grantModel.find({             postal\_code: userPostal,                 location: {                  $near: {                     $maxDistance: 4000,                     $geometry: {                         type: "Point",                         coordinates: parentCoordinates                     }                 }             }}, function(err,result){             if (err) throw err;             console.log(result.length);             var eligibility = result.length;             var cost = 0;             if (eligibility == 1){                 if (marital == "single"){                 cost = 10000;                 }                 if (marital == "married"){                    cost = 20000                 }                 res.send({'eligibility': 1, 'cost':cost});             }             else{                 res.send({'eligibility': 0});             }         });     }); |
| Add bookmark | INSERT INTO  user\_bookmarks\_location  (uid, lid, bookmark\_name)  VALUES  (  (SELECT id FROM User WHERE  username = '"+globalusername+"'  LIMIT 1),  "+ locationId+",  '"+block+"'  ) | bookmarkModel.findOneAndUpdate(     { username: globalusername },     { $push: {         resid\_bookmarks: postalCode     }}, function(err, result){         if (err) throw err;         console.log("Bookmarked!");         res.send(result);     }); |
| Get bookmarks | SELECT latitude, longitude FROM location L WHERE L.id ="+locationId+"  SELECT ub.\* FROM user\_bookmarks\_location ub, user u WHERE u.id = ub.uid AND u.username = '"+globalusername+"' | bookmarkModel.aggregate([         {$unwind: "$resid\_bookmarks"},{ $lookup: { from:"residential", localField:"resid\_bookmarks", foreignField:"postal\_code", as:"bookmark\_details" }}     ]).exec((err, result) => {         if (err) throw err;         res.send(result);         console.log(result);     }) |
| Delete bookmark | DELETE FROM user\_bookmarks\_location WHERE bookmark\_name = '"+bookmarkId+"' | bookmarkModel.update(     {username: globalusername},     { $pull: {         resid\_bookmarks: bookmarkId     }}, function(err,result){         if (err) throw err;         console.log("Deleted!");         res.send(result);     }) |

Technologies used: Mongoose, ExpressJS, Leaflet, NodeJS, EJS,

db.school.find({location:{$geoWithin:{$centerSphere:[[103.8082549, 1.27736919],1/6378.1]}}})

db.user.aggregate([ {$unwind: "$resid\_bookmarks"},{ $lookup: { from:"residential", localField:"resid\_bookmarks", foreignField:"postal\_code", as:"bookmark\_details" }} ])

“SELECT DISTINCT longitude, latitude FROM housing WHERE postal\_code = ‘100088’”

"SELECT \*, "+

   "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

   "AS `distance` "+

   "FROM housing WHERE sh.level = 'Primary' "+

   "WHERE postal\_code = '' "+

   "HAVING `distance` < "+proximity+" " +

   "ORDER BY `distance` ASC; ";

var proximity = 2; //km

var latitude = 1.277369190;

var longitude = 103.8082549;

  // let primaryQuery = 'SELECT \* FROM school WHERE level = "primary"' based on distance;

  let primaryQuery = "SELECT sh.\*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM school\_distance sh WHERE sh.level = 'Primary' "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";

  let secondaryQuery = "SELECT sh.\*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM school\_distance sh WHERE sh.level = 'Secondary' "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";

  let combinedQuery = "SELECT sh.\*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM school\_distance sh WHERE sh.level <> 'Secondary' AND sh.level <> 'Primary' "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";

  let busStopQuery = "SELECT bs.\*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM bus\_stop\_distance bs "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";

  let nppQuery = "SELECT \*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM npc\_distance WHERE npc\_name LIKE '%post%' "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";

  let npcQuery = "SELECT \*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM npc\_distance WHERE npc\_name NOT LIKE '%post%' "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";

  let hawkerQuery = "SELECT \*, "+

  "(6371 \* acos(cos( radians ("+latitude+" ) ) \* cos( radians( `latitude` ) ) \* cos(radians( `longitude` ) - radians("+longitude+")) + sin(radians("+latitude+")) \* sin(radians(`latitude`)))) "+

  "AS `distance` "+

  "FROM hawker\_distance "+

  "HAVING `distance` < "+proximity+" " +

  "ORDER BY `distance` ASC; ";