**List of precomputed tables:**

```
DROP TABLE IF EXISTS states CASCADE;
DROP TABLE IF EXISTS users CASCADE;
DROP TABLE IF EXISTS categories CASCADE;
DROP TABLE IF EXISTS products CASCADE;
DROP TABLE IF EXISTS orders CASCADE;

CREATE TABLE states (
   id    SERIAL PRIMARY KEY,
   name  TEXT NOT NULL UNIQUE
);

CREATE TABLE users (
   id    SERIAL PRIMARY KEY,
   name  char(50) NOT NULL UNIQUE,
   role  char(8) NOT NULL,
   age   INTEGER NOT NULL,
   state_id INTEGER REFERENCES states (id) NOT NULL
);

CREATE TABLE categories (
   id  SERIAL PRIMARY KEY,
   name  char(50) NOT NULL UNIQUE,
   description  char(50) NOT NULL
);

CREATE TABLE products (
   id SERIAL PRIMARY KEY,
   name char(50) NOT NULL,
   sku CHAR(10) NOT NULL UNIQUE,
   category_id INTEGER REFERENCES categories (id) NOT NULL,
   price FLOAT NOT NULL CHECK (price >= 0),
   is_delete BOOLEAN NOT NULL
);

CREATE TABLE orders (
   id SERIAL PRIMARY KEY,
   user_id INTEGER REFERENCES users (id) NOT NULL,
   product_id INTEGER REFERENCES products (id) NOT NULL,
   quantity INTEGER NOT NULL,
   price FLOAT NOT NULL CHECK (price >= 0),
   is_cart BOOLEAN NOT NULL
```

```
);



-- similar products indices
CREATE INDEX index_orders_productid ON orders(product_id);
CREATE INDEX index_orders_userid ON orders(user_id);
CREATE INDEX index_users_stateid ON users(state_id);
```

**Precomputation code:**

```
  var query = 'DROP TABLE IF EXISTS state_headers CASCADE; ' +

            'CREATE TABLE state_headers( ' +
            '  id          SERIAL PRIMARY KEY, ' +
            '  state       TEXT NOT NULL, ' +
            '  state_id    INTEGER REFERENCES states (id) NOT NULL, ' +
            '  category_id INTEGER REFERENCES categories (id) NOT NULL, ' +
            '  total       FLOAT NOT NULL ' +
            '); ' +

            'INSERT INTO state_headers(state, state_id, category_id, total) ' +
            '(SELECT s.name, s.id, p.category_id AS category_id,
COALESCE(SUM(o.price*o.quantity), 0) AS total ' +
            'FROM states s ' +
            'LEFT OUTER JOIN users u ON u.state_id = s.id ' +
            'LEFT OUTER JOIN orders o ON u.id = o.user_id ' +
            'INNER JOIN products p ON p.id = o.product_id ' +
            'GROUP BY s.id, s.name, category_id); ' +

            'DROP TABLE IF EXISTS product_headers CASCADE; ' +

            'CREATE TABLE product_headers( ' +
            '  id          SERIAL PRIMARY KEY, ' +
            '  product     TEXT NOT NULL, ' +
            '  product_id  INTEGER REFERENCES products (id) NOT NULL, ' +
            '  category_id INTEGER REFERENCES categories (id) NOT NULL, ' +
            '  total       FLOAT NOT NULL ' +
```

```
        '); ' +

        'INSERT INTO product_headers(product, product_id, category_id, total) ' +
        '(SELECT p.name AS name, p.id AS product_id, p.category_id AS category_id,
COALESCE(SUM(s.price*s.quantity), 0) AS total ' +
        'FROM products p ' +
        'LEFT OUTER JOIN orders s ON p.id = s.product_id ' +
        'GROUP BY p.id, p.name, p.category_id ' +
        'ORDER BY total DESC); ' +

        'DROP TABLE IF EXISTS analytics CASCADE; ' +

        'CREATE TABLE analytics( ' +
        '  id          SERIAL PRIMARY KEY, ' +
        '  state_id    INTEGER REFERENCES states (id) NOT NULL, ' +
        '  product_id  INTEGER REFERENCES products (id) NOT NULL, ' +
        '  total       FLOAT NOT NULL ' +
        '); ' +

        'INSERT INTO analytics(state_id, product_id, total) ' +
        '(SELECT s.id, p.id, COALESCE(SUM(o.price*o.quantity), 0) AS total ' +
        'FROM states s ' +
        'CROSS JOIN products p ' +
        'LEFT OUTER JOIN users u ON u.state_id = s.id ' +
        'LEFT OUTER JOIN orders o ON o.product_id = p.id AND o.user_id = u.id ' +
        'GROUP BY s.id, p.id ' +
        'ORDER BY s.id); ' +

        'CREATE INDEX index_product ON analytics(product_id); ' +
        'CREATE INDEX index_state ON analytics(state_id); ';
```

**Code that takes care of the buying:**

```
//TODO: make sure this works
createOrders = function(num_orders, done){

  db.any("SELECT count(*) FROM orders")
    .then( function (old_total) {
```

```
    var random_num = Math.random() * 30 + 1;
    if (num_orders < random_num) random_num = num_orders;

    console.log(num_orders);
    console.log(random_num);

    var query = "SELECT proc_insert_orders(" + parseInt(num_orders) + "," +
parseInt(random_num) + ")";

    db.any(query)
      .then( function (data) {
        var query2 = "INSERT INTO new_orders" +
                    "(user_id, product_id, quantity, price, is_cart)" +
                    "(SELECT user_id, product_id, quantity, price, is_cart" +
                    " FROM orders WHERE id > " + old_total[0].count + ")";

        db.any(query2)
          .then( function (data) {
            console.log(data);
            done(true);
          })
          .catch(function (error) {
            console.log(error);
            done(false);
        });

       })
       .catch(function (error) {
         console.log(error);
         done(false);
    });

   })
   .catch(function (error) {
     console.log(error);
     done(false);
 });

}
```

**Code that executes on run:**

```
ctlr.getData = function(category_id){
    console.log(category_id);
    var href = 'http://localhost:3000/api/headers?category=' + category_id;
    $http.get(href)
        .success(function(data, status, headers, config){
            console.log("success");
            console.log(data);
            ctlr.cols = data.products.length;
            ctlr.products = data.products;
            ctlr.states = data.states;
            ctlr.cells = data.cells;
            ctlr.buildTable();
        })
        .error(function(data, status, headers, config){
            console.log("failed");
        });
}
```

```
router.get('/headers/', function(req, res, next){
        console.log("run query request");
        db.update(function(success){
                if (success){
                        console.log("precomputed tables updated");
                        db.getHeaders(req.query.category, function(product_headers,
state_headers, success){
                                if (success){
                                        console.log("headers retreived");
                                        var body = {
                                                products: product_headers,
                                                states: state_headers
                                        }
                                        var cells = [[], [], [], [], [], [], [], [], [], [],
                                                        [], [], [], [], [], [], [], [], [], [],
                                                        [], [], [], [], [], [], [], [], [], [],
                                                        [], [], [], [], [], [], [], [], [], [],
                                                        [], [], [], [], [], [], [], [], [], []];
```

```
                                        getCells(cells, 0, 0, body, function(cells){
                                                console.log("cells retreived");
                                                body.cells = cells;
                                                res.json(body);
                                        });
                                }
                                else {
                                        res.json({"error":"error"});
                                }
                        })
                }
                else{
                        res.json({"error":"error"});
                }
        })
})


update = function(done){

var query = 'UPDATE state_headers x ' +
        'SET total= (x.total + y.total) ' +
        'FROM ' +
        '(SELECT s.name AS state, s.id AS state_id, p.category_id AS category_id,
COALESCE(SUM(o.price*o.quantity), 0) AS total ' +
        'FROM new_orders o, states s, users u, products p ' +
        'WHERE u.state_id = s.id ' +
        'AND p.id = o.product_id ' +
        'AND u.id = o.user_id ' +
        'GROUP BY s.id, s.name, category_id) y ' +
        'WHERE x.state_id = y.state_id ' +
        'AND x.category_id = y.category_id; ' +

        'UPDATE product_headers x ' +
        'SET total = (x.total + y.total) ' +
        'FROM ' +
        '(SELECT p.name AS product, p.id AS product_id, p.category_id AS category_id,
COALESCE(SUM(o.price*o.quantity), 0) AS total ' +
        'FROM new_orders o, products p ' +
        'WHERE p.id = o.product_id ' +
        'GROUP BY p.id, p.name, p.category_id ' +
        'ORDER BY total DESC) y ' +
        'WHERE x.category_id = y.category_id ' +
```

```
        'AND x.product_id = y.product_id; ' +

        'UPDATE analytics x ' +
        'SET total = (x.total + y.total) ' +
        'FROM ' +
        '(SELECT s.id AS state_id, p.id AS product_id, COALESCE(SUM(o.price*o.quantity), 0)
AS total ' +
        'FROM new_orders o, products p, states s, users u ' +
        'WHERE o.product_id = p.id ' +
        'AND o.user_id = u.id ' +
        'AND u.state_id = s.id ' +
        'GROUP BY s.id, p.id ' +
        'ORDER BY s.id) y ' +
        'WHERE x.state_id = y.state_id ' +
        'AND x.product_id = y.product_id; ' +

        'DELETE FROM new_orders *;';

  //console.log(query);

  db.any(query)
    .then(function (data) {
      console.log(data);
      done(true);
    })
    .catch(function (error) {
      console.log(error);
      done(false);
  });

}



getHeaders = function(category, done){

  if (category != 'all'){
    var query1 = "SELECT * FROM product_headers " +
                 "WHERE category_id = " + category + " " +
                 "ORDER BY total DESC " +
                 "LIMIT 50;";

    var query2 = "SELECT y.name as state, y.id as state_id, COALESCE(total, 0) as total " +
```

```
            "FROM " +
             "(SELECT state, state_id, SUM(total) as total  " +
             "FROM state_headers  " +
            "WHERE category_id = " + category + " " +
             "GROUP BY state, state_id  " +
             "ORDER BY total DESC, state ASC) x " +
             "FULL OUTER JOIN " +
             "(SELECT * FROM states) y " +
             "ON x.state = y.name " +
             "AND x.state_id = y.id " +
             "ORDER BY total DESC;";
}
else {
  var query1 = "SELECT product, product_id, SUM(total) as total FROM product_headers " +
            "GROUP BY product, product_id " +
            "ORDER BY total DESC " +
            "LIMIT 50;";

  var query2 = "SELECT y.name as state, y.id as state_id, COALESCE(total, 0) as total " +
            "FROM " +
             "(SELECT state, state_id, SUM(total) as total  " +
             "FROM state_headers  " +
             "GROUP BY state, state_id  " +
             "ORDER BY total DESC, state ASC) x " +
             "FULL OUTER JOIN " +
             "(SELECT * FROM states) y " +
             "ON x.state = y.name " +
             "AND x.state_id = y.id " +
             "ORDER BY total DESC;";
}

console.log(query2);


db.any(query1)
  .then(function (product_headers) {

    db.any(query2)
      .then(function (state_headers) {

        done(product_headers, state_headers, true);
      })
      .catch(function (error) {
```

```
        console.log(error);
        done(null, null, false);
      });

    })
    .catch(function (error) {
      console.log(error);
      done(null, null, false);
  });

}



var getCells = function(cells, p, s, body, done){
          if (p == body.products.length && s == body.states.length)
       done(cells);
    else {

       if (p == body.products.length) p = 0;
       if (s == body.states.length) s = 0;

       var state = body.states[s];
       var product = body.products[p];

       db.getCell(product.product_id, state.state_id, function(result, success){
          if (success){
                  cells[s].push(result[0]);
                  p++;
             if (p == body.products.length) s++;
             getCells(cells, p, s, body, done);
          }
          else {
                  done(cells);
          }
       })
    }
}



getCell = function(product_id, state_id, done){
  var query = "SELECT * FROM analytics " +
```

```
            "WHERE state_id = " + state_id + " " +
            "AND product_id = " + product_id;

  db.any(query)
    .then(function (result) {
      done(result, true);
    })
    .catch(function (error) {
      console.log(error);
      done(null, false);
  });

}


          ctlr.buildTable = function(){

              var tablehead = "<table><thead><tr><td>     Products ><br>States
\/ </td>";
              var table = "</tr></thead><tbody>";
              var tablefoot = "</tbody>"

              ctlr.width = 0;
              if (ctlr.cols < 50) ctlr.width = (ctlr.cols + 1) * 100;
              else ctlr.width = 5100;

              for (var i = 0; i < ctlr.products.length; i++){
                  var product = ctlr.products[i];
                  tablehead += "<td id='ph" + product.product_id + "' data-price=" +
parseFloat(product.total) + "><b>" + product.product + " ($" + rounding(product.total) +
")</b></td>";
              }

              for (var i = 0; i < ctlr.states.length; i++){
                  var state = ctlr.states[i];
                  if (ctlr.products.length > 0){
                      table += "</tr><tr id='state" + state.state_id + "'>";
                      table += "<td class='state_header' data-price=" + parseFloat(state.total) + "><b>"
+ state.state + " ($" + rounding(state.total) + ")</b></td>";

                      for (var j = 0; j < ctlr.products.length; j++){
                          var product = ctlr.products[j];
```

```
                    table += "<td data-price=" + parseFloat(ctlr.cells[i][j].total) + " id='product" +
product.product_id + "'>" + rounding(ctlr.cells[i][j].total) + "</td>";
                }

            }
        }

        var complete = tablehead + table + tablefoot;
        document.getElementById("table").innerHTML = complete;

    }
```

**Code that executes on refresh**

```
    ctlr.refresh = function(){
        var category_id = $('#categorySelect').val();
        var href = 'http://localhost:3000/api/refresh?category=' + category_id;
        $http.get(href)
            .success(function(data, status, headers, config){
                console.log("success");
                console.log(data);
                ctlr.updateTable(data);
            })
            .error(function(data, status, headers, config){
                console.log("failed");
            });

    }
```

```
router.get('/refresh/', function(req, res, next){
    console.log("refresh table request");
    console.log(req.query.category);
    db.updateStuff(req.query.category, function(results, success){
        if (success){
            res.json(results);
        }
        else{
            res.json({"error":"error"});
```

```
                }
        })

})



updateStuff = function(category, done){

  var query1 = "SELECT s.name AS state, s.id AS state_id,
COALESCE(SUM(o.price*o.quantity), 0) AS total " +
            "FROM states s, users u, new_orders o, products p " +
            "WHERE u.state_id = s.id " +
            "AND u.id = o.user_id " +
            "AND p.id = o.product_id ";
  if (category != 'all') query1 += "AND p.category_id = " + category + " ";
  query1 +=    "GROUP BY s.name, s.id " +
            "ORDER BY total DESC"

  var query2 = "SELECT p.name AS product, p.id AS product_id,
COALESCE(SUM(o.price*o.quantity), 0) AS total " +
            "FROM products p, new_orders o " +
            "WHERE p.id = o.product_id ";
  if (category != 'all') query2 += "AND p.category_id = " + category + " ";
  query2 +=    "GROUP BY p.id, p.name " +
            "ORDER BY total DESC ";

  var query3 = "SELECT s.id AS state_id, p.id AS product_id,
COALESCE(SUM(o.price*o.quantity), 0) AS total " +
            "FROM new_orders o, states s, products p, users u " +
            "WHERE u.state_id = s.id " +
            "AND o.product_id = p.id " +
            "AND o.user_id = u.id ";
  if (category != 'all') query3 += "AND p.category_id = " + category + " ";
  query3 +=    "GROUP BY s.id, p.id " +
            "ORDER BY s.id";


  if (category != 'all'){
    var query4 = "SELECT * FROM product_headers " +
              "WHERE category_id = " + category + " " +
              "ORDER BY total DESC " +
              "LIMIT 50;";
```

```
}
else {
  var query4 = "SELECT product, product_id, SUM(total) as total FROM product_headers " +
               "GROUP BY product, product_id " +
               "ORDER BY total DESC " +
               "LIMIT 50;";
}




console.log(query1);

db.any(query1)
  .then(function (states) {

    db.any(query2)
      .then(function (products) {

        db.any(query3)
          .then(function (cells) {

            db.any(query4)
              .then(function (top50) {

                var body = {
                  state_headers: states,
                  product_headers: products,
                  cells: cells,
                  top50: top50
                }

                done(body, true);

              })
              .catch(function (error) {
                console.log(error);
                done(null, false);
            });


          })
          .catch(function (error) {
```

```
            console.log(error);
            done(null, false);
          });

        })
        .catch(function (error) {
          console.log(error);
          done(null, false);
      });

    })
    .catch(function (error) {
      console.log(error);
      done(null, false);
  });

}



 ctlr.updateTable = function(data){

            for (var i = 0; i < data.state_headers.length; i++){
                var state = data.state_headers[i];
                var old_total = $('#state' + state.state_id + ' > .state_header').attr("data-price");
                var new_total = rounding(parseFloat(old_total) + parseFloat(state.total));
                $('#state' + state.state_id + ' > .state_header').addClass('updated_cell');
                $('#state' + state.state_id + ' > .state_header').text("" + state.state + " ($" +
new_total + ")");
            }

            for (var i = 0; i < data.product_headers.length; i++){
                var product = data.product_headers[i];
                var old_total = $('#ph' + product.product_id).attr("data-price");
                var new_total = rounding(parseFloat(old_total) + parseFloat(product.total));
                $('#ph' + product.product_id).addClass('updated_cell');
                $('#ph' + product.product_id).text("" + product.product + " ($" + new_total + ")");
            }

            for (var i = 0; i < data.cells.length; i++){
                var cell = data.cells[i];
                var selector = '#state' + cell.state_id + ' > #product' + cell.product_id;
```

```
        var old_total = $(selector).attr("data-price");
        var new_total = rounding(parseFloat(old_total) + parseFloat(cell.total));
        $(selector).addClass('red_cell');
        $(selector).text("" + new_total);
    }

    var newtop = data.top50;
    var oldtop = ctlr.products;

    var matching = [];

    for (var i = 0; i < newtop.length; i++){
        for (var j = i; j < oldtop.length; j++){
            var p1 = newtop[i];
            var p2 = oldtop[j];
            if (p1.product_id == p2.product_id){
                break;
            }
            if (j == oldtop.length - 1){
                var id = p2.product_id;
                $('#ph' + id).addClass("purple_cell");
                $('#product' + id).addClass("purple_cell");
            }
        }
    }

}
```