# Reference Laboratory Project - Section 6: SQL Queries

## Personal Project by Ryan Breen

### Original Work performed 9/1/2022

The following section utilizes SQL queries known as the *Data Manipulation Langauge (DML)* to create resultant rows and scalars that provide quintessential information needed to be used by the enterprise to perform day-to-day functions as well as serving as a basis to provide business insights for development and guidance.

Like most ideas in life, the queries have been organized by the chief entity that they modify. For example if we want to create a query to look at charges by geographic area - the statement is filed under the charges DML sub-section; however, this organizational scheme could have followed the other direction.

Note, in reality these queries would each be scripted as `.sql` files that could be elicited using any number of coding languages not limited to even command prompts or bash commands.

Finally, large tables have been limited to only 5 rows just to be able to output as a pdf file. Simply delete the limit 5 command and all rows will appear. Also, these statements can be used in any SQL scripting editor and do not need Python per se. Python has been used simply to place statements in a Jupyter notebook. Note, these `SQL` statements have all been created using the *mock laboratory database* I originally created in the first sections using a `Postgres` database with a `PGAdmin4` GUI editor.

## Import Necessary Libraries:

`psycopg2` is the adapter used with `Postgres` databases with `Python` . Below `Pandas` is used as the 'face' of `Python` .

In [1]:
```
import os
import psycopg2 as ps
import pandas as pd
```

## Install Dependencies:

(ipython-sql allows running queries in notebook)

In [ ]:
```
#pip install ipython-sql
```

## Load `IPython-SQL` module:

In [1]:
```
%load_ext sql
```

## Create a Connection:

A connection is made here to a local `Postgres` database (127.0.0.1) where the port is defaulted to 5432. Database name is Cap-Sensitive.

In [3]:
```
%sql postgresql://postgres:7009@localhost/Lab_Project
```

### Test Connection with Simple Query:

In [4]:
```
%%sql
select *
from Customers
limit 1
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1 rows affected.
```

Out[4]:

| customer_id | address_id | dob | gender | race |
|---|---|---|---|---|
| 10000000 | 10000000 | 2015-07-08 | Female | Hispanic |

## Custom Function to Convert SQL Queries to .csv Files:

In [5]:
```
path = os.getcwd()
csv_=path+'\\csv\\'
def sql_csv(file):
    result = _
    df = result.DataFrame()
    df.to_csv(csv_+f'{file}.csv')
```

## Below Each Entity with Relevant DML Queries:

Below, each `SQL` query is set to a `Python` string using triple quotes (avoids conflicts with literal quotes). The strings (queries) alongside the database connection information are then able to be introduced into the read_sql_query method that convets the database table into a pandas dataframe. Once achieved, the dataframe can then be manipulated in multiple ways such as producing visualizations, .csv files, and performing *object-oriented programming* using `Python` .

### Customers:

Information for Marketting

Customers Ages Binned by Proportions:

In [6]:
```
%%sql
with table1 as
(select
(case when foo.age < 18 then '0 - 18'
when foo.age >= 18 and foo.age < 30 then '18 - 35'
when foo.age >= 30 and foo.age < 55 then '30 - 55'
when foo.age >= 55 and foo.age < 75 then '55 - 75'
when foo.age >= 75 then '75 - 100'
end) as ages,
count(foo.dob)
from
(select (extract(year from now())-extract(year from dob)) as age,
 dob
 from customers) as foo
 group by ages
 order by ages),
 table2 as
 (select count(customer_id) as total
 from customers)
 select ages,
 cast(table1.count as numeric)/cast(table2.total as numeric) as prop
 from table1, table2
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[6]:

| ages | prop |
|---|---|
| 0 - 18 | 0.15164000000000000000 |
| 18 - 35 | 0.12360000000000000000 |
| 30 - 55 | 0.24610000000000000000 |
| 55 - 75 | 0.20150000000000000000 |
| 75 - 100 | 0.27716000000000000000 |

In [7]:
```
sql_csv('cust_ages_bin_prop')
```

Customers Gender Binned by Proportions:

In [8]:
```
%%sql
with table1 as
(select cast(count(customer_id) as numeric) as total
from customers),
table2 as
(select gender,
cast(count(customer_id) as numeric) as counts
from customers
group by gender)
select table2.gender,
round(table2.counts/table1.total,3)as props
from table1, table2
```

```
 * postgresql://postgres:***@localhost/Lab_Project
2 rows affected.
```

Out[8]:

| gender | props |
|---|---|
| Female | 0.497 |
| Male | 0.503 |

In [9]:
```
sql_csv('cust_gender_bin_prop')
```

Customers Race Binned by Proportions:

In [10]:
```
%%sql
with table1 as
(select cast(count(customer_id) as numeric) as total
from customers),
table2 as
(select race,
cast(count(customer_id) as numeric) as counts
from customers
group by race)
select table2.race,
```

```
round(table2.counts/table1.total,3)as props
from table1, table2
```

```
 * postgresql://postgres:***@localhost/Lab_Project
4 rows affected.
```

Out[10]:

| race | props |
|---|---|
| Hispanic | 0.246 |
| Caucasian | 0.251 |
| Asian | 0.254 |
| African American | 0.249 |

In [11]:

```
sql_csv('cust_race_bin_prop')
```

### Customer Surveys:

Customer Survey Averages:

In [12]:

```
%%sql
select avg(courteous) as avg_courteous,
avg(schedule) as avg_schedule,
avg(costs) as avg_costs,
avg(delivery) as avg_delivery,
avg(overall) as avg_overall
from customer_surveys;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1 rows affected.
```

Out[12]:

| avg_courteous | avg_schedule | avg_costs | avg_delivery | avg_overall |
|---|---|---|---|---|
| 7.0076400000000000 | 6.9979200000000000 | 6.9999200000000000 | 6.9986000000000000 | 6.9987200000000000 |

In [13]:

```
sql_csv('cust_surveys_avg')
```

Customer Survey Monthly Averages:

In [4]:

```
%%sql
select extract(month from customer_surveys.dates) as month,
avg(courteous) as avg_courteous,
avg(schedule) as avg_schedule,
avg(costs) as avg_costs,
avg(delivery) as avg_delivery,
avg(overall) as avg_overall
from customer_surveys
group by extract(month from customer_surveys.dates)
order by month
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[4]:

| month | avg_courteous | avg_schedule | avg_costs | avg_delivery | avg_overall |
|---|---|---|---|---|---|
| 1 | 5.9437984496124031 | 5.3628875968992248 | 6.4040697674418605 | 5.6816860465116279 | 5.0489341085271318 |
| 2 | 5.8629782833505688 | 5.5315408479834540 | 6.5511892450879007 | 5.6494312306101344 | 4.9994829369183040 |
| 3 | 6.0295748613678373 | 5.5281885397412200 | 6.4353049907578558 | 5.6529574861367837 | 5.0485212569316081 |
| 4 | 6.0196642685851319 | 5.5035971223021583 | 6.5860911270983213 | 5.7218225419664269 | 4.9904076738609113 |
| 5 | 5.9669187145557656 | 5.5179584120982987 | 6.4404536862003781 | 5.7273156899810964 | 5.1148393194706994 |

In [15]:

```
sql_csv('cust_surveys_avg_monthly')
```

### Employees:

Information for Human Resources

Employees Ages Binned by Proportions:

In [16]:

```
%%sql
with table1 as
(select
(case when foo.age < 18 then '0 - 18'
when foo.age >= 18 and foo.age < 30 then '18 - 35'
when foo.age >= 30 and foo.age < 55 then '30 - 55'
when foo.age >= 55 and foo.age < 75 then '55 - 75'
when foo.age >= 75 then '75 - 100'
```

```
end) as ages,
count(foo.dob)
from
(select (extract(year from now())-extract(year from dob)) as age,
 dob
 from employees) as foo
 group by ages
 order by ages),
 table2 as
 (select count(employee_id) as total
 from employees)
 select ages,
 cast(table1.count as numeric)/cast(table2.total as numeric) as prop
 from table1, table2
```

```
 * postgresql://postgres:***@localhost/Lab_Project
3 rows affected.
```

Out[16]:

| ages | prop |
|---|---|
| 0 - 18 | 0.42000000000000000000 |
| 18 - 35 | 0.25000000000000000000 |
| 30 - 55 | 0.33000000000000000000 |

In [17]:

```
sql_csv('employee_ages_bin_prop')
```

Employees Gender Binned by Proportions:

In [18]:

```
%%sql
with table1 as
(select cast(count(employee_id) as numeric) as total
from employees),
table2 as
(select gender,
cast(count(employee_id) as numeric) as counts
from employees
group by gender)
select table2.gender,
round(table2.counts/table1.total,3)as props
from table1, table2
```

```
 * postgresql://postgres:***@localhost/Lab_Project
2 rows affected.
```

Out[18]:

| gender | props |
|---|---|
| Female | 0.730 |
| Male | 0.270 |

In [19]:

```
sql_csv('employee_gender_bin_prop')
```

Employees Absolute Genders :

In [20]:

```
%%sql
select gender,
count(employee_id)
from employees
group by gender;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
2 rows affected.
```

Out[20]:

| gender | count |
|---|---|
| Female | 73 |
| Male | 27 |

In [21]:

```
sql_csv('employee_gender_bin')
```

Employees Race Binned by Proportions:

In [22]:

```
%%sql
with table1 as
(select cast(count(employee_id) as numeric) as total
from employees),
table2 as
(select race,
cast(count(employee_id) as numeric) as counts
from employees
```

```
group by race)
select table2.race,
round(table2.counts/table1.total,3)as props
from table1, table2
```

```
 * postgresql://postgres:***@localhost/Lab_Project
4 rows affected.
```

Out[22]:

| race | props |
|---|---|
| Hispanic | 0.080 |
| Asian | 0.180 |
| Caucasian | 0.500 |
| African American | 0.240 |

In [23]:

```
sql_csv('employee_race_bin_prop')
```

Employee Absolute Race:

In [24]:

```
%%sql
select race,
count(employee_id)
from employees
group by race;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
4 rows affected.
```

Out[24]:

| race | count |
|---|---|
| Hispanic | 8 |
| Asian | 18 |
| Caucasian | 50 |
| African American | 24 |

In [25]:

```
sql_csv('employee_gender_bin')
```

## Employee Surveys:

Information for Human Resources

In [5]:

```
%%sql
select employee_surveys.employee_id,
avg(pay),
avg(manager),
avg(work_volume),
avg(available_tools),
avg(overall)
from employee_surveys
group by employee_surveys.employee_id
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[5]:

| employee_id | avg | avg_1 | avg_2 | avg_3 | avg_4 |
|---|---|---|---|---|---|
| 10000023 | 7.0000000000000000 | 7.6666666666666667 | 6.7500000000000000 | 7.0833333333333333 | 6.9166666666666667 |
| 10000093 | 6.6666666666666667 | 8.1666666666666667 | 7.3333333333333333 | 6.8333333333333333 | 6.5000000000000000 |
| 10000095 | 6.8333333333333333 | 7.6666666666666667 | 7.0000000000000000 | 7.0833333333333333 | 7.0833333333333333 |
| 10000077 | 7.0833333333333333 | 7.6666666666666667 | 7.1666666666666667 | 6.5000000000000000 | 6.5833333333333333 |
| 10000096 | 6.9166666666666667 | 7.9166666666666667 | 6.7500000000000000 | 6.9166666666666667 | 6.5000000000000000 |

In [27]:

```
sql_csv('employee_survey_avg_monthly')
```

## Finances:

Net Profit:

In [28]:

```
%%sql
with overhead as
(select sum(salary) as overhead
from employees),
expenses as
```

```
(select sum(electric + water  + waste) as expenses
from expenses),
revenue as
(select sum(cast(cost as numeric)) as revenue
from orders
left outer join panels
on orders.panel_id = panels.panel_id)
select (revenue.revenue - (overhead.overhead+expenses.expenses)) as Net_Profit
from revenue, overhead, expenses
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1 rows affected.
```

Out[28]:

| net_profit |
|---|
| 30787972.10 |

In [29]:

```
sql_csv('finances_net_profit')
```

Utilities:

In [30]:

```
%%sql
select (sum(electric) + sum(water) + sum(waste)) as utilities
from expenses;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1 rows affected.
```

Out[30]:

| utilities |
|---|
| 84257.21 |

In [31]:

```
sql_csv('finances_costs_utilities')
```

Revenue:

In [ ]:

```
%%sql
select sum(cast(panels.cost as numeric)) as Revenue
from panels
left outer join orders
on panels.panel_id = orders.panel_id;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
```

In [33]:

```
sql_csv('finances_gains_revenue')
```

Inventory Costs:

In [34]:

```
%%sql
select sum(costs) as deliveries
from shipments;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1 rows affected.
```

Out[34]:

| deliveries |
|---|
| 36425.28 |

In [42]:

```
sql_csv('finances_costs_deliveries')
```

Monthly Revenue by Lab:

In [6]:

```
%%sql
select extract(month from orders.date) as month,
laboratories.names as names,
sum(panels.cost) as cost
from orders
left outer join laboratories
on orders.lab_id = laboratories.lab_id
left outer join panels
on orders.panel_id = panels.panel_id
group by month, names
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[6]:

| month | names | cost |
|---|---|---|
| 1 | Central Lab | 616182.45 |

| | | |
|---|---|---|
| 1 | Downtown Lab | 620131.12 |
| 1 | East Lab | 618404.12 |
| 1 | Main Lab | 613248.77 |
| 1 | North Lab | 611146.17 |

In [7]:

```
sql_csv('revenue_monthly_lab')
```

### Customer Testing:

Customer Results: **Large Memory!**

In [43]:

```
%%sql
select orders.order_id, orders.customer_id, orders.panel,
orders.date, orders.time, test_definitions.tests,
test_definitions.mean, test_definitions.units
from orders
left outer join test_definitions
on orders.panel_id = test_definitions.panel_id
and orders.lab_id = test_definitions.lab_id;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5598358 rows affected.
```

In [44]:

```
sql_csv('customer_results')
```

Customer Results within 95% CI:

In [7]:

```
%%sql
select foo.lab_id,
foo.tests,
(foo.actual_mean - foo.target_mean)/(foo.actual_sd/sqrt(foo.n)) as t_score,
case
when (foo.actual_mean - foo.target_mean)/(foo.actual_sd/sqrt(foo.n)) > 1.96
or (foo.actual_mean - foo.target_mean)/(foo.actual_sd/sqrt(foo.n)) < -1.96
then 'outlier'
else 'OK'
END CI_95
from
(select patient_results.lab_id,
test_definitions.tests as tests,
avg(patient_results.results) as actual_mean,
stddev(patient_results.results) as actual_sd,
test_definitions.mean as target_mean,
test_definitions.sd as target_sd,
count(test_definitions.tests) as n
from patient_results
left outer join test_definitions
on patient_results.test_definition_id = test_definitions.test_definition_id
and patient_results.lab_id = test_definitions.lab_id
group by patient_results.lab_id,
test_definitions.tests,
test_definitions.mean,
test_definitions.sd) as foo
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[7]:

| lab_id | tests | t_score | ci_95 |
|---|---|---|---|
| 100 | ALBUMIN | -2.1426809146881705 | outlier |
| 100 | ALT | -0.5711304850445698 | OK |
| 100 | APTT | 0.6446354458992181 | OK |
| 100 | AST | -0.6211556201422935 | OK |
| 100 | BICARBONATE | -0.7606828629582597 | OK |

In [52]:

```
sql_csv('customer_results_outliers')
```

Customer Result Descriptives:

In [8]:

```
%%sql
select test_definitions.tests,
round(avg(patient_results.results),2) as mean,
round(stddev(patient_results.results),2) as sd,
round(stddev(patient_results.results) / avg(patient_results.results),2) as cv,
round(avg(patient_results.results) - stddev(patient_results.results)*3,2) as ThreeSDLow,
```

```
round(avg(patient_results.results) + stddev(patient_results.results)*3,2) as ThreeSDHi,
min(patient_results.results),
max(patient_results.results)
from patient_results
left outer join test_definitions
on patient_results.test_definition_id = test_definitions.test_definition_id
group by test_definitions.tests
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[8]:

| tests | mean | sd | cv | threesdlow | threesdhi | min | max |
|---|---|---|---|---|---|---|---|
| ALBUMIN | 4.14 | 0.23 | 0.06 | 3.45 | 4.83 | 3.16 | 5.29 |
| ALT | 23.66 | 1.32 | 0.06 | 19.70 | 27.62 | 17.88 | 29.81 |
| APTT | 15.78 | 0.88 | 0.06 | 13.14 | 18.42 | 12.24 | 19.33 |
| AST | 21.69 | 1.21 | 0.06 | 18.07 | 25.31 | 16.19 | 27.03 |
| BICARBONATE | 24.65 | 1.37 | 0.06 | 20.54 | 28.76 | 18.69 | 30.62 |

In [60]:

```
sql_csv('customer_results_desc')
```

### Orders (Billing): **Large Memory**!:

Charge per Order with Customer:

In [ ]:

```
%%sql
select panels.panel, panels.cost, orders.lab_id, orders.customer_id,
orders.date, orders.time
from panels
left outer join orders
on panels.panel_id = orders.panel_id;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1000000 rows affected.
```

Query to Be Handled by BI Software due to large size of file.

Charges by Date:

In [21]:

```
%%sql
select orders.date,
sum(cast(panels.cost as numeric))
from panels
left outer join orders
on panels.panel_id = orders.panel_id
group by orders.date
having orders.date = '2021-01-01';
```

```
 * postgresql://postgres:***@localhost/Lab_Project
1 rows affected.
```

Out[21]:

| date | sum |
|---|---|
| 2021-01-01 | 99019.16 |

Charges by Gender:

In [63]:

```
%%sql
select customers.gender,
sum(cast(panels.cost as numeric)),
avg(cast(panels.cost as numeric))
from panels
left outer join orders
on panels.panel_id = orders.panel_id
left outer join customers
on orders.customer_id = customers.customer_id
group by customers.gender;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
2 rows affected.
```

Out[63]:

| gender | sum | avg |
|---|---|---|
| Female | 17962901.02 | 36.1409127527040948 |
| Male | 18184299.61 | 36.1534140992810790 |

In [64]:

```
sql_csv('charges_by_gender')
```

Charges by Binned Ages:

In [65]:

```
%%sql
select bar.age_groups,
sum(cost)
from
(select
    CASE WHEN foo.years > 0 and foo.years <= 20 THEN '0-20'
         WHEN foo.years > 20 and foo.years <= 40 THEN '20-40'
         WHEN foo.years > 40 and foo.years <= 60 THEN '40-60'
         WHEN foo.years > 60 and foo.years <= 80 THEN '60-80'
         ELSE '> 80' END
         AS age_groups,
         foo.cost
from
(select
  ((current_date - customers.dob)/365) as years,
 cast(panels.cost as numeric)
from customers
left outer join orders
on orders.customer_id = customers.customer_id
left outer join panels
on orders.panel_id = panels.panel_id) as foo) as bar
group by bar.age_groups;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[65]:

| age_groups | sum |
|---|---|
| > 80 | 7812182.36 |
| 20-40 | 7254611.75 |
| 40-60 | 7163919.99 |
| 60-80 | 7254358.11 |
| 0-20 | 6662128.42 |

In [66]:

```
sql_csv('charges_by_ages_bin')
```

### Geography:

Data to be used in Cluster Analysis

Customers by Geography:

In [9]:

```
%%sql
select customer_id, lat, lon
from customers
left outer join addresses
on customers.address_id = addresses.address_id
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[9]:

| customer_id | lat | lon |
|---|---|---|
| 10000000 | 48.0517637 | -122.1770818 |
| 10000001 | 33.7825194 | -117.2286478 |
| 10000002 | 41.49932 | -81.6943605 |
| 10000003 | 42.262593200000005 | -71.8022934 |
| 10000004 | 34.0007104 | -81.0348144 |

In [68]:

```
sql_csv('cust_by_geo')
```

Employees by Geography:

In [10]:

```
%%sql
select employee_id, lat, lon
from employees
left outer join addresses
on employees.address_id = addresses.address_id
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[10]:

| employee_id | lat | lon |
|---|---|---|
| 10000000 | 48.0517637 | -122.1770818 |

| | | |
|---|---|---|
| 10000001 | 33.7825194 | -117.2286478 |
| 10000002 | 41.49932 | -81.6943605 |
| 10000003 | 42.262593200000005 | -71.8022934 |
| 10000004 | 34.0007104 | -81.0348144 |

In [70]:
```
sql_csv('employee_by_geo')
```

Laboratories by Geography:

In [71]:
```
%%sql
select lab_id, lat, lon
from laboratories;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[71]:

| lab_id | lat | lon |
|---|---|---|
| 100 | 48.0517637 | -122.1770818 |
| 101 | 33.7825194 | -117.2286478 |
| 102 | 41.49932 | -81.6943605 |
| 103 | 42.262593200000005 | -71.8022934 |
| 104 | 34.0007104 | -81.0348144 |

In [72]:
```
sql_csv('lab_by_geo')
```

### Business Functions:

Query to Be Handled by BI Software due to large size of file.

Missing Tests: Should be *no* rows indicating no missing test

In [11]:
```
%%sql
select *
from orders
left outer join patient_results
on orders.order_id = patient_results.order_id
where patient_results = NULL;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
0 rows affected.
```

Out[11]:

| order_id | lab_id | customer_id | panel | panel_id | date | time | test_result_id | order_id_1 | test_definition_id | date_1 | time_1 | customer_id_1 | lab_id_1 | panel_1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

In [ ]:
```
```

Daily Quality Control Results:**Large Memory!**

In [13]:
```
%%sql
select date(datetime),lab_id, level, analytes,
mean + ((-4*random())+2)*sd as result, units
from qc_definitions
left outer join qc_results
on qc_results.qc_definition_id = qc_definitions.qc_definition_id
order by date, level
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[13]:

| date | lab_id | level | analytes | result | units |
|---|---|---|---|---|---|
| 2021-01-01 | 100 | 1 | UREA NITROGEN | 13.227657280165142 | mg/dL |
| 2021-01-01 | 100 | 1 | CALCIUM | 8.38175075272051 | mg/dL |
| 2021-01-01 | 100 | 1 | SODIUM | 125.84432034640184 | mmol/L |
| 2021-01-01 | 100 | 1 | PROTEIN | 6.198000986953294 | g/dL |
| 2021-01-01 | 100 | 1 | GGT | 71.52791715397156 | U/L |

Query to Be Handled by BI Software due to large size of file.

Comparing Quality Control Results vs. Customer Testing: **Large Memory!**

- Something causing duplicate results to occurr
- i.e. Five LDL tests alone for lab 100 level

In [15]:

```
%%sql
with qc as
(select date(datetime),lab_id, level, analytes,
mean + ((-4*random())+2)*sd as result, units
from qc_definitions
left outer join qc_results
on qc_results.qc_definition_id = qc_definitions.qc_definition_id
order by date, level),
testing as
(select date, lab_id, tests, avg(results), units
from patient_results
group by date, lab_id, tests, units
order by date, lab_id)
select *
from qc
left outer join testing
on qc.date = testing.date and
qc.lab_id = testing.lab_id and
qc.analytes = testing.tests
order by qc.date, qc.lab_id, qc.level
limit 5;
```

```
 * postgresql://postgres:***@localhost/Lab_Project
5 rows affected.
```

Out[15]:

| date | lab_id | level | analytes | result | units | date_1 | lab_id_1 | tests | avg | units_1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2021-01-01 | 100 | 1 | ALBUMIN | 3.62152151345629 | g/dL | 2021-01-01 | 100 | ALBUMIN | 4.2427819548872180 | g/dL |
| 2021-01-01 | 100 | 1 | ALBUMIN | 3.797122961185859 | g/dL | 2021-01-01 | 100 | ALBUMIN | 4.2427819548872180 | g/dL |
| 2021-01-01 | 100 | 1 | ALBUMIN | 3.8094001578302117 | g/dL | 2021-01-01 | 100 | ALBUMIN | 4.2427819548872180 | g/dL |
| 2021-01-01 | 100 | 1 | ALBUMIN | 3.67176754640724 | g/dL | 2021-01-01 | 100 | ALBUMIN | 4.2427819548872180 | g/dL |
| 2021-01-01 | 100 | 1 | ALBUMIN | 3.9740832866615596 | g/dL | 2021-01-01 | 100 | ALBUMIN | 4.2427819548872180 | g/dL |

Query to Be Handled by BI Software due to large size of file.

## End of Section