From Database to Dashboard Project - Microsoft Power BI

Creating Insights from a Mock Laboratory Database System

Ryan Breen

September 21, 2022

Background

Presenting data by the use of interactive visualizations appears to be second nature for enterprise in terms of gaining insights to help steer such enterprise towards success. The data from original created for the *Mock Laboratory Database* was used (_see Database_to_Dashboard*SQLQueries*) by Microsoft Power BI and transformed specificall for each specific dashboard's needs.

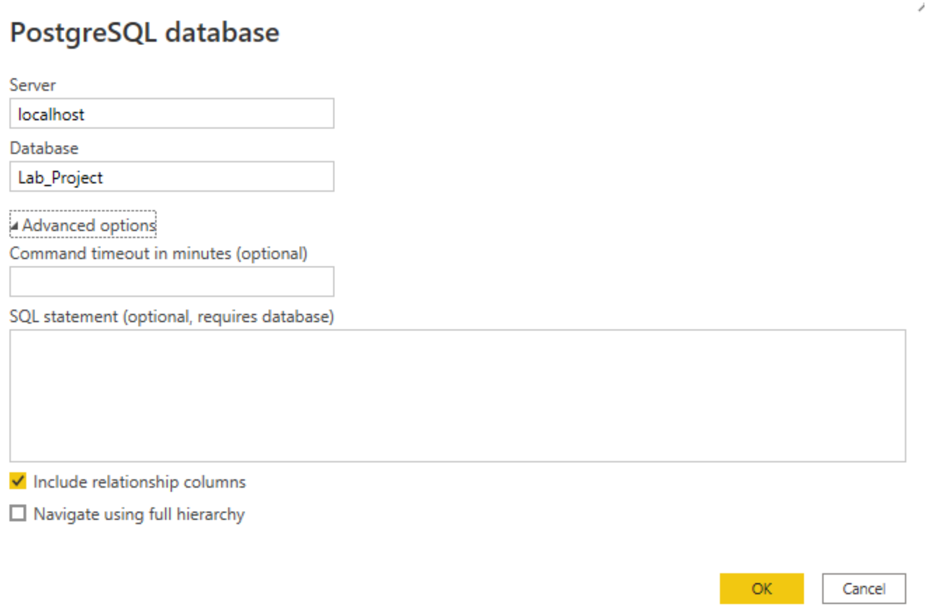
Purpose

In this notebook, the steps taken in regards to the formation of the data transforms and visualizations are highlighted to demonstrate my capabilities to work with the Microsoft Power BI software as well as highlight the softwares capabilities.

Raw Data

Obtaining the data for use in Power BI is simple. You can upload flat files (csv, json, xml, ect) or you may connect to a cloud, internal, or no-SQL database source. Here, a local PostgreSQL database storing data that is used by a "Mock Laboratory" (see previous project notebook) is uploaded using the server and database name. Also, the port 5432 and my personal password are entered in the next screen, but this is not shown for obvious reason..

The data and Power BI dashboards have been created using data from the entire year of 2021. This brief but hopefully insightful presentation is suppose to mimic a year report representing five major categories within the Mock Laboratory Company. Note, this company has five subsidary laboratories across the United States that each form the larger enterprise. Data from each and the enterprise as a whole are impactful and have been created below.


PostgreSQL database

Server: localhost

Database: Lab_Project

Advanced options

Command timeout in minutes (optional):

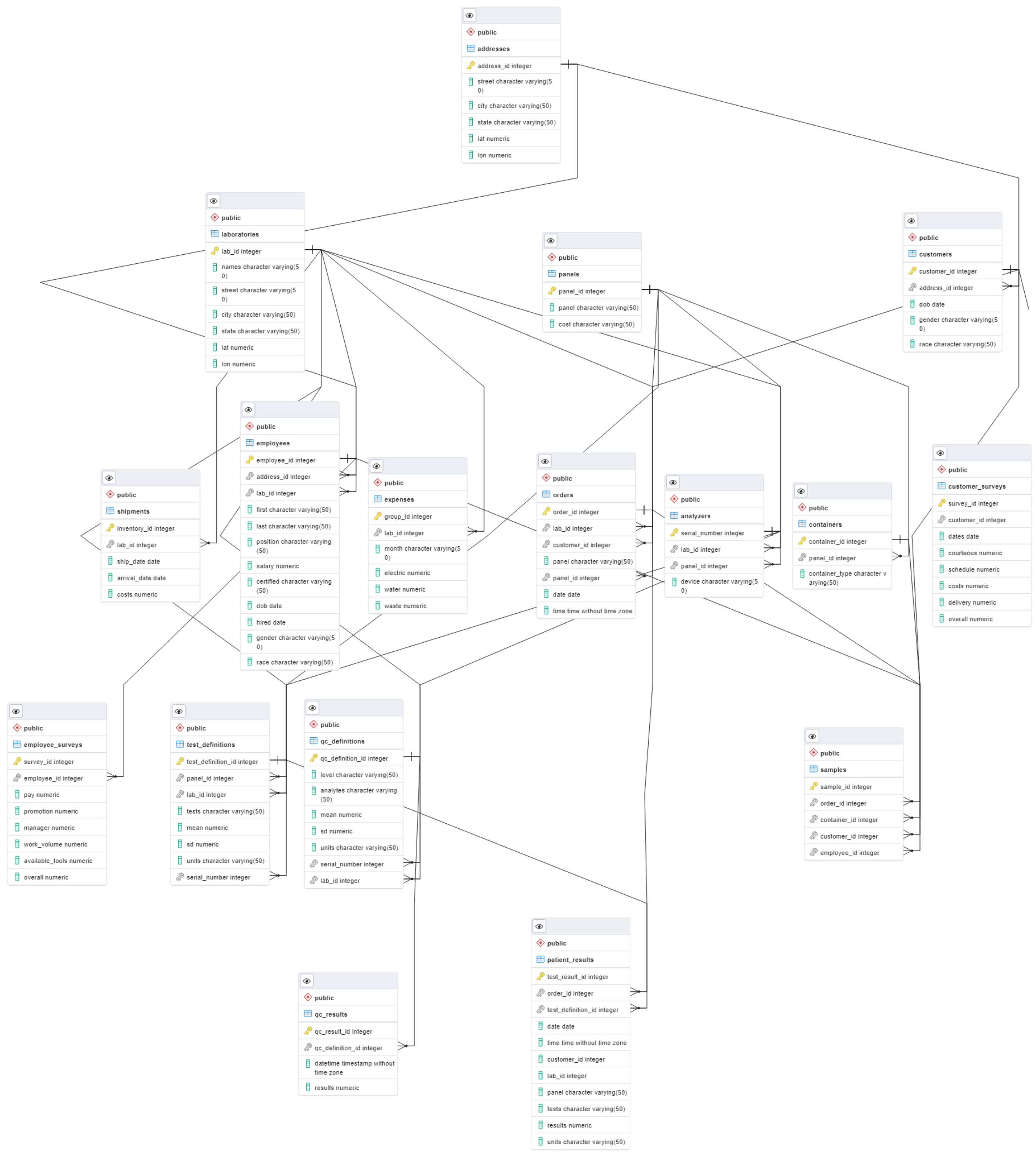
SQL statement (optional, requires database):

☑ Include relationship columns

☐ Navigate using full hierarchy

About the Database

Below is the logical schema for the database that may be referenced to gain knowledge of table-column and table-table relationships throughout this project



Organization of the Dashboards

Again, 5 different laboratories comprise up of the entire business where customers pay and have laboratory testing performed at one of the five laboratories that provide blood test results. Customers order panels that are simply a bunch of related test in one orderable group. The money, results, customer, employee, and overall volume are all tracked in the Mock Laboratory to be elicited by Power BI to create dashboards like the one outlined just below:

Dashboards have been created to suffice five major business components in this particular presentation. Below, the information needed and the tables used are given:

1. Accounting - Where is the company in terms of making money?
2. Customers - Who are the customers?
3. Employees - Who works here? Is the company practicing diversity?
4. Orders
5. References - measures the quality of each test

The Data Transformations with Perspective Dashboards

The steps taken in the Power BI Power Query and Data Transformation sections of the software are first given to demonstrate what data cleaning/transformation steps where needed to create each dashboard. Second, the a print screen of the actual dashboard is given and then the link to the actual dashboard follows to allow users the ability to interactively use each filter.

Accounting Dashboard

- a. Profit-YTD-All Labs (Revenue - (Expenses + Overhead)
  - calculated from b - d below
- b. Revenue-YTD-All Labs (money coming in from selling lab tests)
  - left join between orders and panels tables
- c. Expenses-YTD-All Labs (cost to "keep the lights on")
  - directly from expenses table
- d. Overhead-YTD-All Labs (cost to pay all employees)
  - directly from the employees table
- e. Monthly breakdown of revenue
  - left join between orders and panels group by months
- f. Filter that can display either or all the different

Accouting Data Cleaning/Transformations:

Two different tables ('Accounting' and 'total_charges') created

Accounting table cleaning/transformations:

In [ ]:

```
Expenses = SUM('public expenses'[water]) + SUM('public expenses'[waste]) + SUM('public expenses'[electric])
Overhead = SUM('public employees'[salary])
Revenue = SUM('orders'[public panels.cost])
Profit = [Revenue] - ([Overhead] + [Expenses])
```

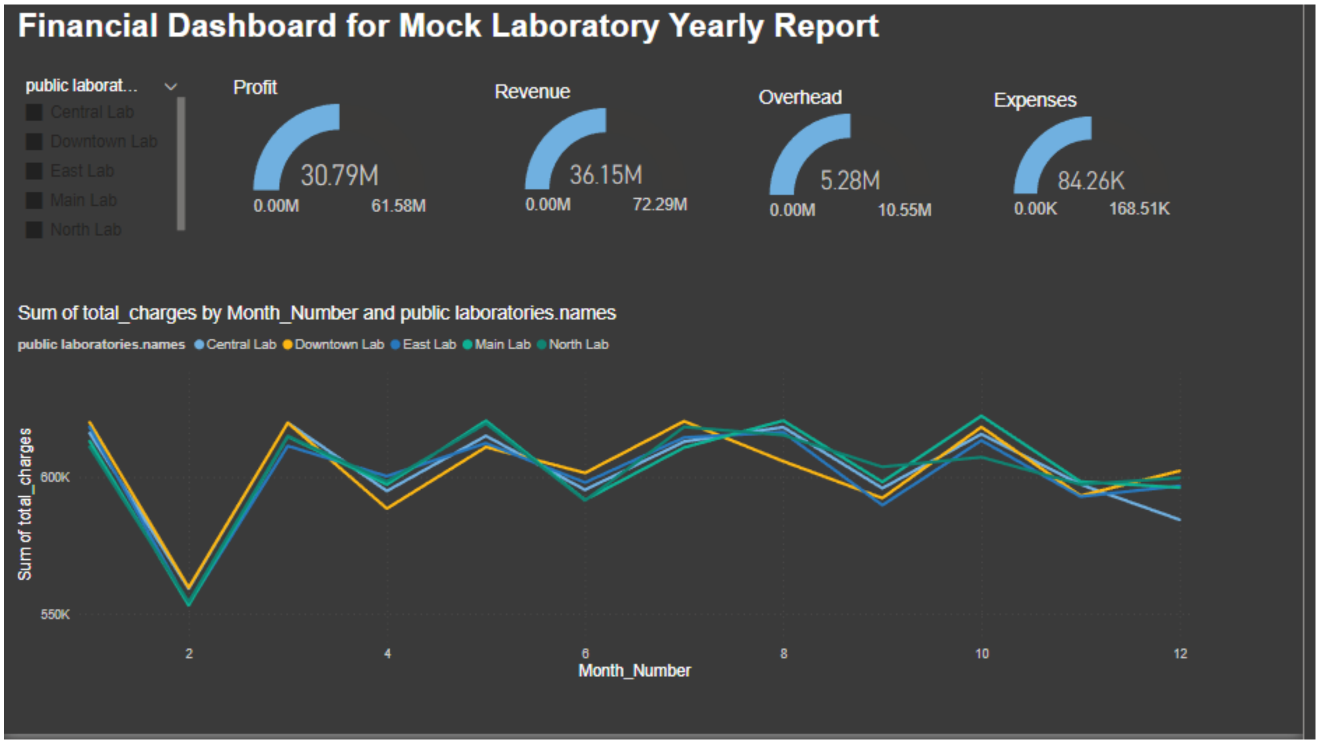
total_charges cleanign/transformations:

In [ ]:

```
# obtain data from data source
= PostgreSQL.Database("localhost", "Lab_Project")
# start of working with orders table (parent table)
= Source{[Schema="public",Item="orders"]}[Data]
# left join on orders and panels table
= Table.NestedJoin(public_orders, {"panel_id"}, #"public panels", {"panel_id"}, "public panels", JoinKind.LeftOuter
# keep just panel.costs column from prior join
= Table.ExpandTableColumn(#"Merged Queries", "public panels", {"cost"}, {"public panels.cost"})
# removes the other suggested joins
= Table.RemoveColumns(#"Expanded public panels",{"public.customers", "public.laboratories"})
# left join orders with laboratories table
= Table.NestedJoin(#"Removed Other Columns", {"lab_id"}, #"public laboratories", {"lab_id"}, "public laboratories",
= Table.SelectColumns(#"Expanded public laboratories",{"date", "public panels.cost", "public laboratories.names"})
= obtain month number by extract month numeral in date
= Table.TransformColumns(#"Removed Other Columns1", {{"date", each Text.BeforeDelimiter(Text.From(_, "en-US"), "/")
# aggregate sum for the cost of all panels ordered
= Table.Group(#"Extracted Text Before Delimiter", {"date", "public laboratories.names"}, {{"total_charges", each L
# create a duplicate month column to create an 'abbrevated month' column
= Table.RemoveColumns(#"Added Conditional Column",{"month"})
# create conditional column where 1 = Jan, 2=Feb, et al.
= Table.AddColumn(#"Removed Columns1", "Custom", each if [date] = "1" then "Jan" else if [date] = "2" then "Feb" el
= Table.RemoveColumns(#"Added Conditional Column1",{"date"})
# Re-create numeral month column (accidentally deleted) to create sortability
= Table.AddColumn(#"Sorted Rows", "Custom", each if [month] = "Jan" then 1 else if [month] = "Feb" then 2 else if [
# Rename column above to month_number
= Table.RenameColumns(#"Added Conditional Column2",{{"Custom", "Month_Number"}})
# finally sort table by month number
= Table.Sort(#"Reordered Columns1",{{"Month_Number", Order.Ascending}})
```

Accounting Dashboard:

The accounting table is simply a group of financial calculations or 'measures' as stated by Power BI. Here is how to create each of the measures.



Inferences Made from Dashboard:

1. The company is doing very well in terms of profitability - Profits account for over 80% of the cash flow
2. The month of February indicated by monthnumber 2 indicates a big dip in revenue - this event should definitely be investigated for causes.

Accounting Interactive Dashboard Link:

In [ ]:

```

```

Customers Dashboard

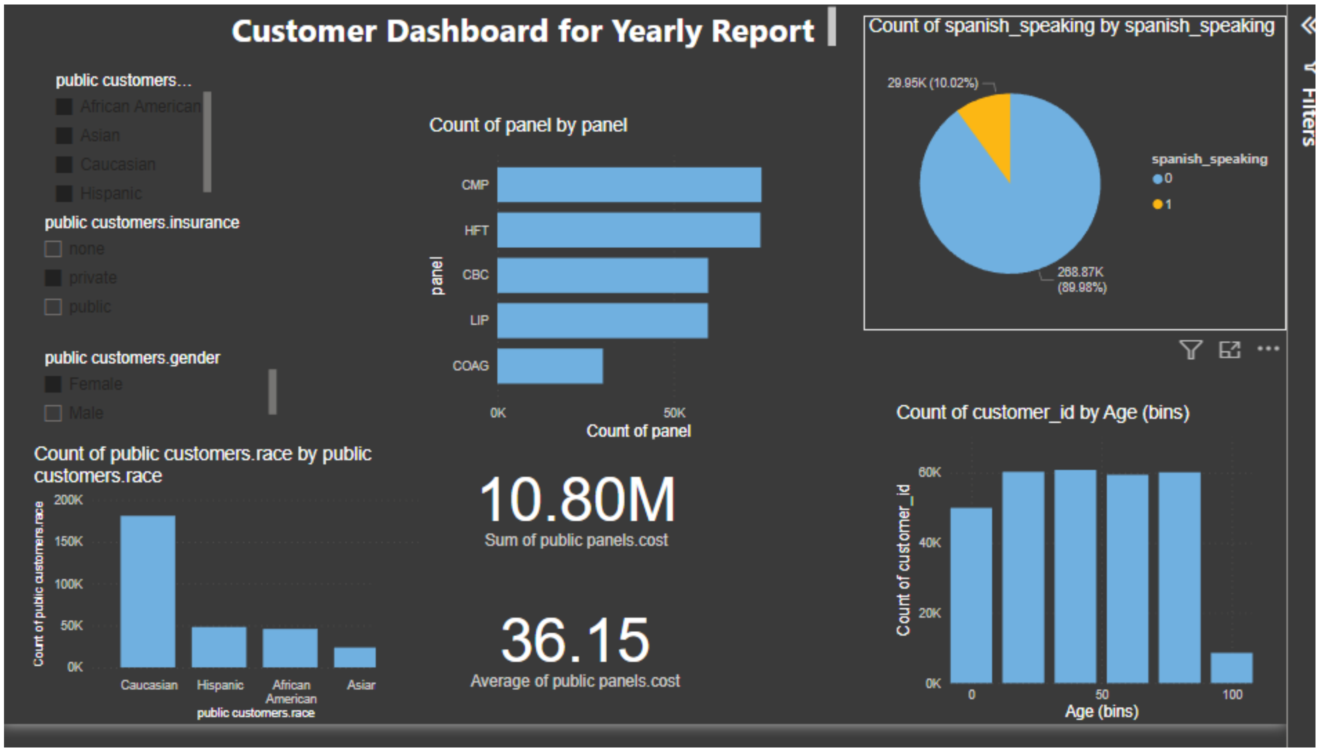
- a. Breakdown of type of panels orderd by customers
- b. Breakdown of gender
- c. Breakdown of racial profiles
- d. Total amount spent by customers
- e. Average amount spent by customers
- f. Binned age profile (20 year bins)
- g. Filter for all above by race, insurance status, and gender.

Customers Data Cleaning/Transformations:

In [ ]:

```
# left join orders and customers tables
= Table.NestedJoin(orders, {"customer_id"}, #"public customers", {"customer_id"}, "public customers", JoinKind.Left
# keep selected columns
= Table.ExpandTableColumn(Source, "public customers", {"dob", "gender", "race", "spanish_lang", "english_only", "ma
# expand out 'suggested join' addresses table with customers table (suggested joins read meta data from relationshi
= Table.ExpandTableColumn(#"Merged Queries", "public customers", {"address_id"}, {"public customers.address_id"})
# left join addresses and customers table
= Table.NestedJoin(#"Expanded public customers1", {"public customers.address_id"}, #"public addresses", {"address_i
# keep selected addresses table columns
= Table.ExpandTableColumn(#"Merged Queries1", "public addresses", {"address_id", "street", "city", "state", "lat",
# binarize (one hot encode) spanish language column from no to 0 and yes to 1
= Table.AddColumn(#"Expanded public addresses", "Custom", each if [public customers.spanish_lang] = "no" then 0 els
= Table.RenameColumns(#"Added Conditional Column",{{"Custom", "spanish_speaking"}})
# binarize (one hot encode) english only column from no to 0 and yes to 1
= Table.AddColumn(#"Renamed Columns", "Custom", each if [public customers.english_only] = "no" then 0 else 1)
= Table.RenameColumns(#"Added Conditional Column1",{{"Custom", "english_only"}})
# binarize (one hot encode) married column from no to 0 and yes to 1
= Table.AddColumn(#"Renamed Columns1", "married", each if [public customers.marital_status] = "no" then 0 else 1)
= Table.AddColumn(#"Added Conditional Column2", "customer_insurance", each if [public customers.insurance] = "priva
# renamed lat to latitude and long to longitude
= Table.RenameColumns(#"Added Conditional Column3",{{"public addresses.lat", "latitude"}, {"public addresses.lon",
```

Customers Dashboard:



Inferences Made from Dashboard:

1. Coag panel seems to be much less popular than other test panels
2. Interesting the average panel price payed by a customer in this instance is $36.15
3. Ages are well represented in the customer base
4. A fairly spanish speaking population exist in the customer base so there should be resources such as spanish speaking customer services operators and other resources available corresponding to the 11% depicted here.
5. Note, for example sake, we hae selected the filter for all races with individuals with private insurance and Female customers only to demonstrate the ability to filter. Please click on the interactive link below to create your own inferences.

Customers Interactive Dashboard Link:

```
In [ ]:
```

Employees Dashboard

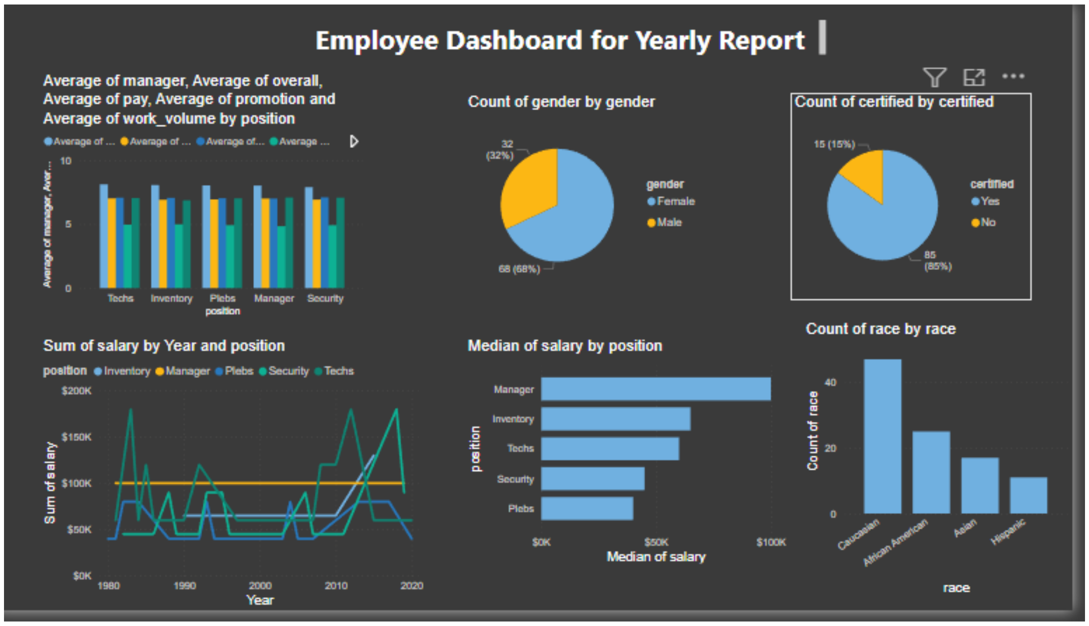
- a. Work volume by employee position by laboratory
  - left join of employee and laboratory and orders
- b. Breakdown of gender
  - directly from employee table
- c. Breakdown of certification (professional certification)
  - directly from employee table
- e. Sum of all salaries per a specific year employee hired
  - directly from employee table
- f. Median salary of employees by position (entire organization)
  - directly from the employee table
- g. Breakdown of racial profile of company to ensure diversity
  - directly from employee table

Employees Data Cleaning/Transformations:

Inferences made directly from employees table

```
In [ ]:
= PostgreSQL.Database("localhost", "Lab_Project")
= Source{[Schema="public",Item="employees"]}[Data]
```

Employees Dashboard:



Inferences Made from Dashboard:

1. Between 1980 and 2020, Inventory had the largest increase in Sum of salary (100.01%) while Techs had the largest decrease (0.01%).
2. The most recent Sum of salary anomaly was in 2012, when Techs had a high of $179,998.41.
3. Sum of salary for Plebs started trending down on 2016, falling by 40,011.07 in 4 years.
4. Across all 5 position, Average of manager ranged from 7.88 to 8.11, Average of overall ranged from 6.90 to 7.01, and Average of pay ranged from 7 to 7.10.
5. Female accounted for 68.00 percent of Count of gender.

Employees Interactive Dashboard Link:

In [ ]:

```

```

Orders Dashboard

Here are the expectations or questions to answer:

- a. Number of orders by specific laboratory - YTD
  - left join of orders and laboratories tables
- b. Total number of orders for 2021
  - directly from orders table
- c. Orders placed on a particular day - helps managers decide schedules
  - directly from orders table
- d. Number of orders by specific panel ordered
  - directly from the orders table
- e. Orders by a specific month - helps determine seasonality in orders
  - directly from the orders table
- f. A filter to discern between different laboratories or test panels
  - directly from the orders table

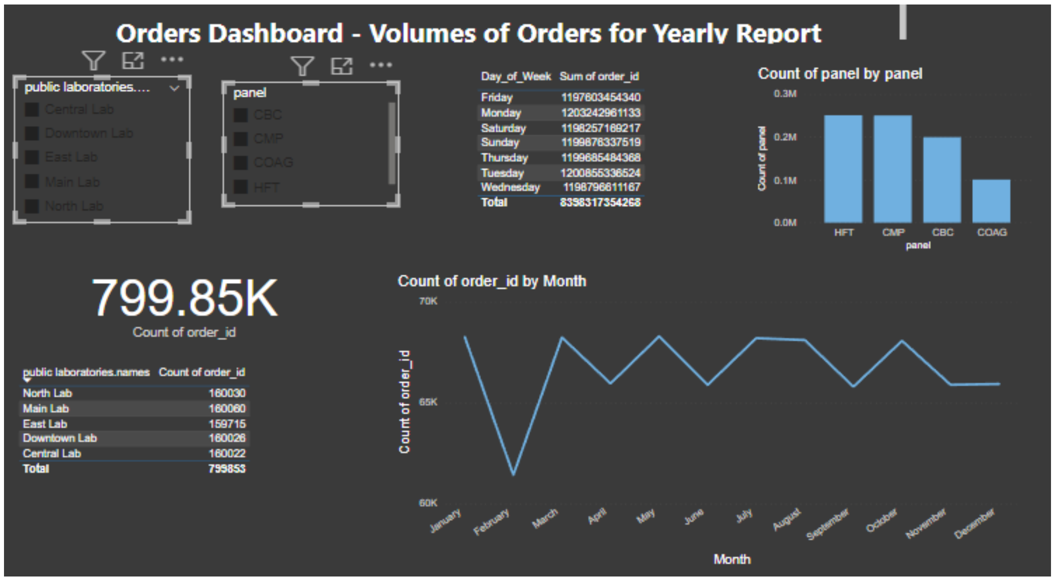
Orders Data Cleaning/Transformations:

In [ ]:

```
# connected to the data source
= PostgreSQL.Database("localhost", "Lab_Project")
# using orders table to start
= Source{[Schema="public",Item="orders"]}[Data]
# left join orders and panels table
= Table.NestedJoin(public_orders, {"panel_id"}, #"public panels", {"panel_id"}, "public panels", JoinKind.LeftOuter
# kept only panels.costs column from previous join
= Table.ExpandTableColumn(#"Merged Queries", "public panels", {"cost"}, {"public panels.cost"})
# removed all other 'suggested' columns
= Table.RemoveColumns(#"Expanded public panels",{"public.customers", "public.laboratories"})
# kept the following columns
= Table.SelectColumns(#"Removed Columns",{"order_id", "lab_id", "customer_id", "panel", "panel_id", "date", "public
# duplicated the dates column to create potential day of week column
= Table.DuplicateColumn(#"Removed Other Columns", "date", "date - Copy")
# left join orders with laboratories to get lab names
```

```
= Table.NestedJoin(#"Duplicated Column", {"lab_id"}, #"public laboratories", {"lab_id"}, "public laboratories", Joi
# simply inserts all columns from laboratories from previous left join
= Table.ExpandTableColumn(#"Merged Queries1", "public laboratories", {"names"}, {"public laboratories.names"})
# converts dates to day of the week (as integer)
= Table.TransformColumns(#"Expanded public laboratories",{{"date - Copy", Date.DayOfWeek, Int64.Type}})
# converts integer day of week to literal day of week
= Table.AddColumn(#"Calculated Day of Week", "Day_of_Week", each if [#"date - Copy"] = 1 then "Monday" else if [#"d
# removed the previous orders.date (copy) column - only used for previous step only
= Table.RemoveColumns(#"Added Conditional Column",{"date - Copy"})
```

Orders Dashboard:



Day_of_Week	Sum of order_id
Friday	1197603454340
Monday	1203242961133
Saturday	1198257169217
Sunday	1199876337519
Thursday	1199685484368
Tuesday	1200855336524
Wednesday	1198796611167
Total	8398317354268

public laboratories.names	Count of order_id
North Lab	160030
Main Lab	160060
East Lab	159715
Downtown Lab	160026
Central Lab	160022
Total	799853

Inferences Made from Dashboard:

1. Days of the week play an ambivalent role in customer order volumes so we can dismiss the theory that orders are made more on weekday
2. The volumes are allocated fairly even at each laboratory - a great outcome
3. Coag panels seem not to be as popular as the other three panels
4. There is a definite pronounce dip in sales this past February so further investigation linking more information should be looked into

Orders Interactive Dashboard Link:

In [ ]:

```

```

Rereferences Dashboard

- a. Count of testing by alerts
  - three types of alerts exist: low, normal, or high where low and high indicate an abonormal test result.
  - test results that fall outside of the (+/-) 2 SD from a target mean are deemed 'abnormal'
  - left join orders with patient_results tables with column transformations
    - b. Comparison of means by each analyzer
    - left join orders, patient_results, and analyzers tables
    - c. Comparison of actual mean/sd vs. target mean/sd by analyte (test)
    - left join orders, patient_results, and test_spcifications tables
    - d. Test average for specific test by each month in 2021
    - left join orders and patient results

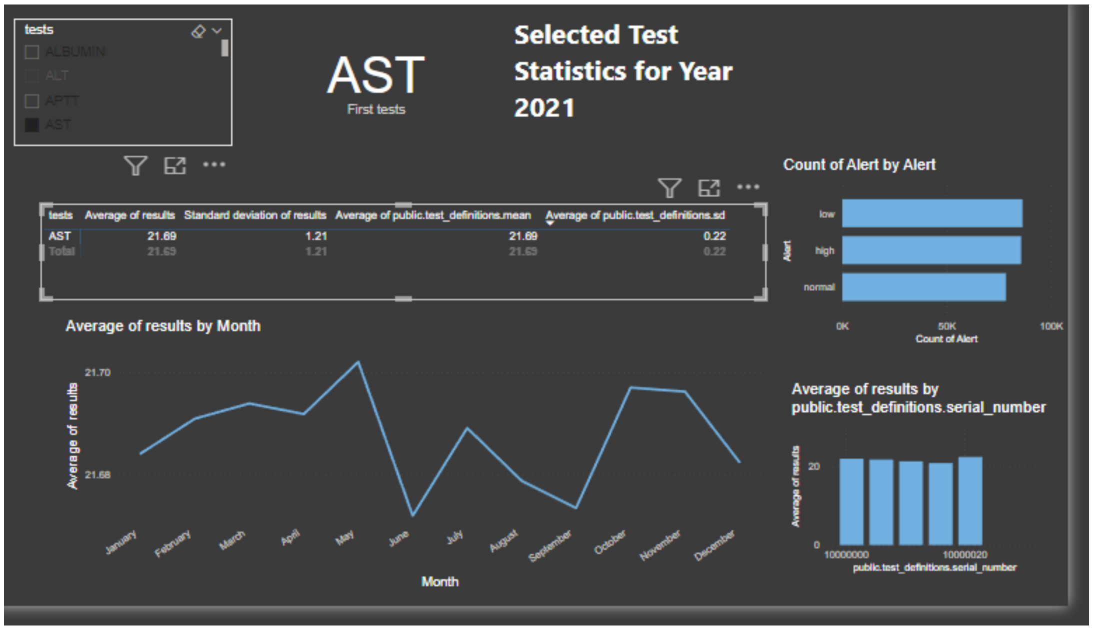
Rereferences Data Cleaning/Transformations:

In [ ]:

```
# connect to the database
= PostgreSQL.Database("localhost", "Lab_Project")
# start off with the primary table patient_results
= Source{[Schema="public",Item="patient_results"]}[Data]
# expand out 'suggested join' analyzer tables
= Table.ExpandRecordColumn(public_patient_results, "public.test_definitions", {"mean", "sd", "serial_number", "publ
= Table.RemoveColumns(#"Expanded public.test_definitions.public.analyzers",{"public.test_definitions.public.laborat
# left join patient_results and laboratories
= Table.NestedJoin(#"Removed Columns1", {"lab_id"}, #"public laboratories", {"lab_id"}, "public laboratories", Join
= Table.ExpandTableColumn(#"Merged Queries", "public laboratories", {"names"}, {"public laboratories.names"})
# create a column that is 2 SD above the target mean
```

```
= Table.AddColumn(#"Expanded public laboratories", "upper_2SD", each [public.test_definitions.mean] + 2 * [public.t
# simply place columns in a logical order
= Table.ReorderColumns(#"Added Custom",{"test_result_id", "order_id", "test_definition_id", "date", "time", "custom
# create a column 2 SD below the target mean
= Table.AddColumn(#"Reordered Columns", "lower_sd", each [public.test_definitions.mean] - 2 * [public.test_definiti
# place last column in logical order
= Table.ReorderColumns(#"Added Custom1",{"test_result_id", "order_id", "test_definition_id", "date", "time", "custo
# rename the newly created columns
= Table.RenameColumns(#"Reordered Columns1",{{"lower_sd", "lower_2sd"}, {"upper_2SD", "upper_2sd"}})
# created a conditional column that deems result 'high' if greater than 2 SD and 'low' if result less than 2 SD aro
= Table.AddColumn(#"Renamed Columns", "Alert", each if [results] < [lower_2sd]then "low" else if [results] > [upper
```

Rereferences Dashboard:



tests	Average of results	Standard deviation of results	Average of public.test_definitions.mean	Average of public.test_definitions.sd
AST	21.69	1.21	21.69	0.22
Total	21.69	1.21	21.69	0.22

Inferences Made from Dashboard:

1. Here the blood test named 'AST' (a liver enzyme) is selected to view various test statistics
2. Right 'right off the bat' we can see that the test average for AST experienced some fluctuation throughout the year with a dip in June.
3. The standard deviation is running 6x greater for the actual results compared to the target standard deviation that is set by the manufacturer of the reagent (using data from their R&D). The test at the lab is experiencing some greater variability. The target and actual mean are running just fine.
4. The bar graph on the bottom right corner indicates how the analyzer runs between all five labs in terms of the mean. The mean does run fairly consistent between all labs.
5. Another inference is made from the alerts bar graph on the middle right that indicates an even number of low, normal, and high results. The low and high alerts are created when values fall outside of a range that is (+/-) 2 SD from the target mean. The target mean is created from the manufactor standard deviation. As stated in stipulation # 3 above, the standard deviation is running 6x greater in actuality then what is purposed by the manufacturer. The range here may be too 'tight' , our analyzer may be too erractic, but in any case we should invesitage this phenomena.

This ends my yearly report for the Mock Laboratory General Performance Report where general but key inferences are made from the dashboard after cleaning / transforming data that is normally housed in the Mock Laboratory Database.