SQL Challenge - Homework 9

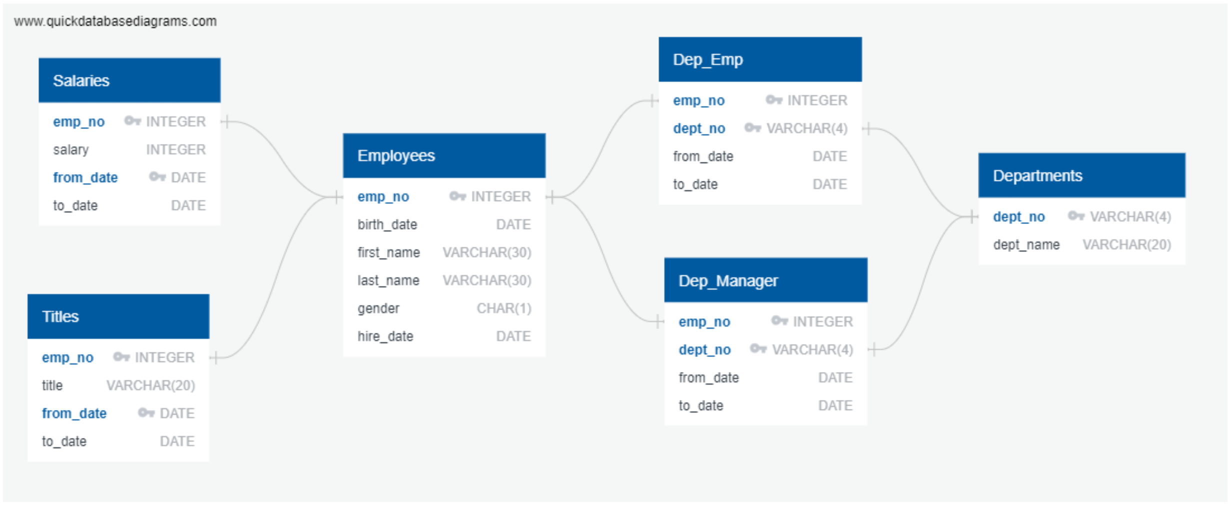
Tools Used

- Postregsql 4 - Type of SQL
- pgAdmin 4 - Database Admin tool
- QuickDBD - Quick Database Diagrams (Tool to create ERD - Entity Relationship Diagrams)
- Jupyter Notebooks - Presentation of work, graphing
- Python, Pandas, MatPlotLib, Numpy, SQL Alchemy - See Imports section

Data Modeling

Using the csv files provided, created an ERD in QuickDBD

- departments.csv
- employees.csv
- salaries.csv
- titles.csv
- dept_emp.csv
- dept_manager.csv



Data Engineering

Next, starting from the ERD above, exported a table schema and edited it for specific data types and other constraints such as primary and foreign keys

Started with this (export from QuickDBD):

```
CREATE TABLE "Departments" (
    "dept_no" VARCHAR(4)   NOT NULL,
    "dept_name" VARCHAR(20)   NOT NULL,
    CONSTRAINT "pk_Departments" PRIMARY KEY (
        "dept_no"
     )
);

CREATE TABLE "Employees" (
    "emp_no" INTEGER   NOT NULL,
    "birth_date" DATE   NOT NULL,
    "first_name" VARCHAR(30)   NOT NULL,
    "last_name" VARCHAR(30)   NOT NULL,
    "gender" CHAR(1)   NOT NULL,
    "hire_date" DATE   NOT NULL,
    CONSTRAINT "pk_Employees" PRIMARY KEY (
        "emp_no"
     )
);

CREATE TABLE "Salaries" (
    "emp_no" INTEGER   NOT NULL,
    "salary" INTEGER   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,
    CONSTRAINT "pk_Salaries" PRIMARY KEY (
        "emp_no","from_date"
     )
);

CREATE TABLE "Titles" (
    "emp_no" INTEGER   NOT NULL,
    "title" VARCHAR(20)   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,
    CONSTRAINT "pk_Titles" PRIMARY KEY (
        "emp_no","from_date"
     )
);

CREATE TABLE "Dep_Emp" (
    "emp_no" INTEGER   NOT NULL,
    "dept_no" VARCHAR(4)   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,
    CONSTRAINT "pk_Dep_Emp" PRIMARY KEY (
        "emp_no","dept_no"
     )
);
```

```
);

CREATE TABLE "Dep_Manager" (
    "emp_no" INTEGER   NOT NULL,
    "dept_no" VARCHAR(4)   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,
    CONSTRAINT "pk_Dep_Manager" PRIMARY KEY (
        "emp_no","dept_no"
     )
);

ALTER TABLE "Salaries" ADD CONSTRAINT "fk_Salaries_emp_no" FOREIGN KEY(
"emp_no")
REFERENCES "Employees" ("emp_no");

ALTER TABLE "Titles" ADD CONSTRAINT "fk_Titles_emp_no" FOREIGN KEY("emp_
no")
REFERENCES "Employees" ("emp_no");

ALTER TABLE "Dep_Emp" ADD CONSTRAINT "fk_Dep_Emp_emp_no" FOREIGN KEY("em
p_no")
REFERENCES "Employees" ("emp_no");

ALTER TABLE "Dep_Emp" ADD CONSTRAINT "fk_Dep_Emp_dept_no" FOREIGN KEY("d
ept_no")
REFERENCES "Departments" ("dept_no");

ALTER TABLE "Dep_Manager" ADD CONSTRAINT "fk_Dep_Manager_emp_no" FOREIGN
KEY("emp_no")
REFERENCES "Employees" ("emp_no");

ALTER TABLE "Dep_Manager" ADD CONSTRAINT "fk_Dep_Manager_dept_no" FOREIG
N KEY("dept_no")
REFERENCES "Departments" ("dept_no");
```

Then updated it to this:

```
CREATE TABLE "Departments" (
    -- Create fields
    "dept_no" VARCHAR(4)  NOT NULL,
    "dept_name" VARCHAR(20) NOT NULL,

    -- Add contstraints
    CONSTRAINT "pk_Departments" PRIMARY KEY ("dept_no")
);

CREATE TABLE "Employees" (
    -- Create fields
    "emp_no" INTEGER   NOT NULL,
    "birth_date" DATE   NOT NULL,
    "first_name" VARCHAR(30)   NOT NULL,
    "last_name" VARCHAR(30)   NOT NULL,
    "gender" CHAR(1)   NOT NULL,
    "hire_date" DATE   NOT NULL,

    -- Add constraints
    CONSTRAINT "pk_Employees" PRIMARY KEY ("emp_no")
);

CREATE TABLE "Salaries" (
    -- Create fields
    "emp_no" INTEGER   NOT NULL,
    "salary" INTEGER   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,

    -- Add constraints
    CONSTRAINT "fk_Salaries_emp_no" FOREIGN KEY("emp_no") REFERENCES "Em
ployees" ("emp_no"),
    CONSTRAINT "pk_Salaries" PRIMARY KEY ("emp_no","from_date")
);

CREATE TABLE "Titles" (
    -- Create fields
    "emp_no" INTEGER   NOT NULL,
    "title" VARCHAR(20)   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,

    -- Add constraints
    CONSTRAINT "fk_Titles_emp_no" FOREIGN KEY("emp_no") REFERENCES "Empl
oyees" ("emp_no"),
    CONSTRAINT "pk_Titles" PRIMARY KEY ("emp_no","from_date")
);

CREATE TABLE "Dep_Emp" (
    -- Create fields
    "emp_no" INTEGER   NOT NULL,
    "dept_no" VARCHAR(4)   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,

    -- Add constraints
    CONSTRAINT "fk_Dep_Emp_emp_no" FOREIGN KEY("emp_no") REFERENCES "Emp
loyees" ("emp_no"),
    CONSTRAINT "fk_Dep_Emp_dept_no" FOREIGN KEY("dept_no") REFERENCES "D
epartments" ("dept_no"),
    CONSTRAINT "pk_Dep_Emp" PRIMARY KEY ("emp_no","dept_no")
);

CREATE TABLE "Dep_Manager" (
    -- Create fields
    "dept_no" VARCHAR(4)   NOT NULL,
    "emp_no" INTEGER   NOT NULL,
    "from_date" DATE   NOT NULL,
    "to_date" DATE   NOT NULL,

    -- Add constraints
    CONSTRAINT "fk_Dep_Manager_dept_no" FOREIGN KEY("dept_no") REFERENCE
S "Departments" ("dept_no"),
    CONSTRAINT "fk_Dep_Manager_emp_no" FOREIGN KEY("emp_no") REFERENCES
 "Employees" ("emp_no"),
    CONSTRAINT "pk_Dep_Manager" PRIMARY KEY ("emp_no","dept_no")
);
```

Using pgAdmin4

- Created the database (SQLChallenge)
- Ran the table schema file above to create the tables
- Imported the csv files into the appropriate tables ensuring that independent tables were run first to avoid conflicts when loading.

Data Analysis

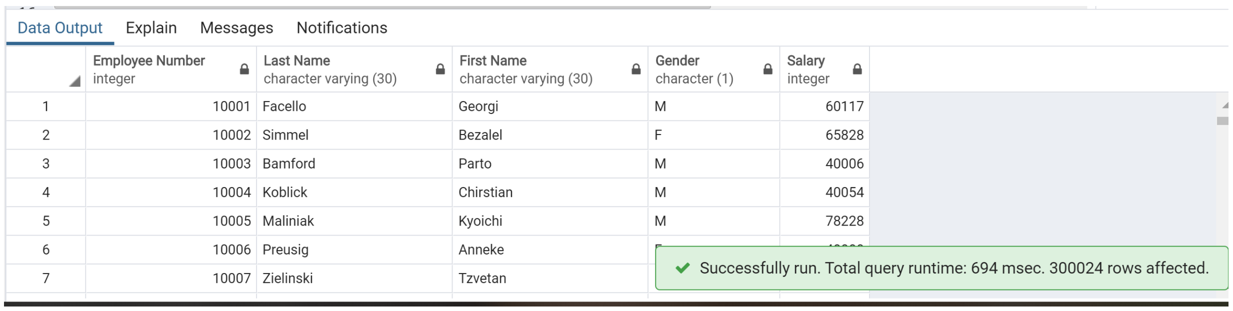
Phase 1 - SQL Queries

- Executed within pgAdmin4

NOTE: Throughout these queries it was unclear if we were to only use current employee data or include historical as well. When in doubt, it was run both ways noting that current employees have a to_date of 9999-01-01

1. List the following details of each employee: employee number, last name, first name, gender, and salary.

```
SELECT emp_no AS "Employee Number", last_name AS "Last Name", first_name
AS "First Name", gender AS "Gender",
    (
        SELECT "Salaries".salary
         FROM "Salaries"
         WHERE "Employees".emp_no = "Salaries".emp_no
    ) AS "Salary"
FROM "Employees";
```

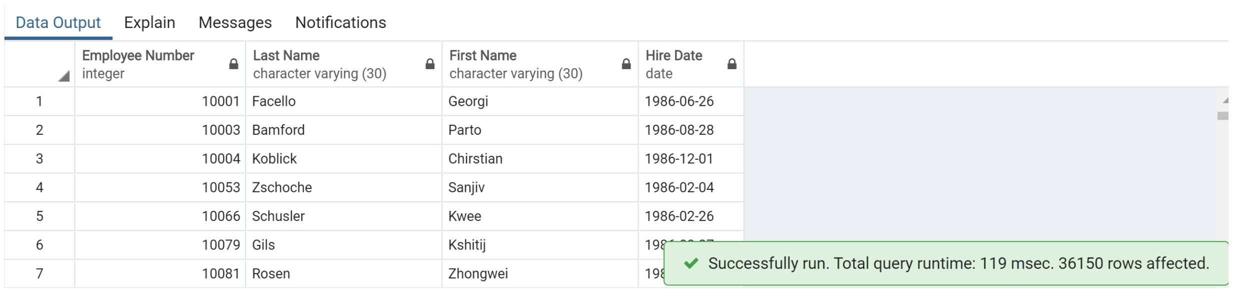

Data Output Explain Messages Notifications

	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Gender character (1)	Salary integer
1	10001	Facello	Georgi	M	60117
2	10002	Simmel	Bezalel	F	65828
3	10003	Bamford	Parto	M	40006
4	10004	Koblick	Chirstian	M	40054
5	10005	Maliniak	Kyoichi	M	78228
6	10006	Preusig	Anneke	[illegible]	[illegible]
7	10007	Zielinski	Tzvetan		

✓ Successfully run. Total query runtime: 694 msec. 300024 rows affected.

2. List employees who were hired in 1986.

```
SELECT emp_no AS "Employee Number", last_name AS "Last Name", first_name
AS "First Name", hire_date AS "Hire Date"
FROM "Employees"
WHERE EXTRACT(year FROM hire_date) = 1986;
```


Data Output Explain Messages Notifications

	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Hire Date date
1	10001	Facello	Georgi	1986-06-26
2	10003	Bamford	Parto	1986-08-28
3	10004	Koblick	Chirstian	1986-12-01
4	10053	Zschoche	Sanjiv	1986-02-04
5	10066	Schusler	Kwee	1986-02-26
6	10079	Gils	Kshitij	[illegible]
7	10081	Rosen	Zhongwei	[illegible]

✓ Successfully run. Total query runtime: 119 msec. 36150 rows affected.

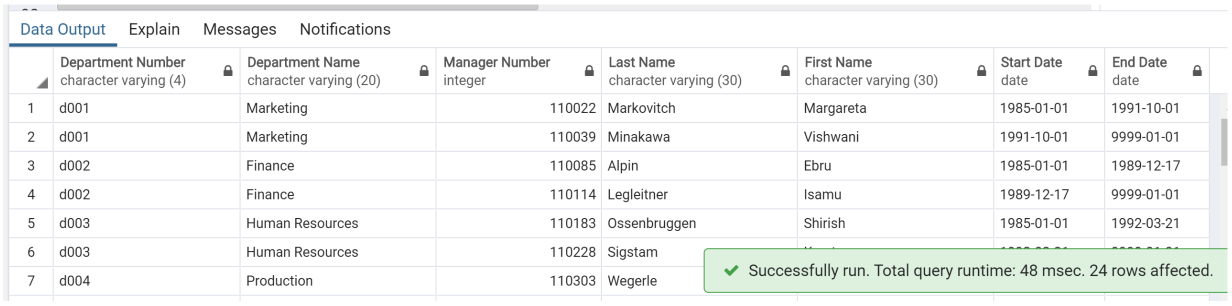
3. List the manager of each department with the following information:

- department number, department name, the manager's employee number, last name, first name, and start and end employment dates.

QUESTION: List the Complete History or just the current managers?

Here is the HISTORY

```
SELECT dpt.dept_no AS "Department Number", dpt.dept_name AS "Department
 Name", dptMgr.emp_no AS "Manager Number",
    emp.last_name AS "Last Name", emp.first_name AS "First Name", dptMg
r.from_date AS "Start Date",
    dptMgr.to_date AS "End Date"
FROM "Departments" AS dpt, "Dep_Manager" AS dptMgr, "Employees" AS emp
WHERE dpt.dept_no = dptMgr.dept_no AND dptMgr.emp_no = emp.emp_no;
```

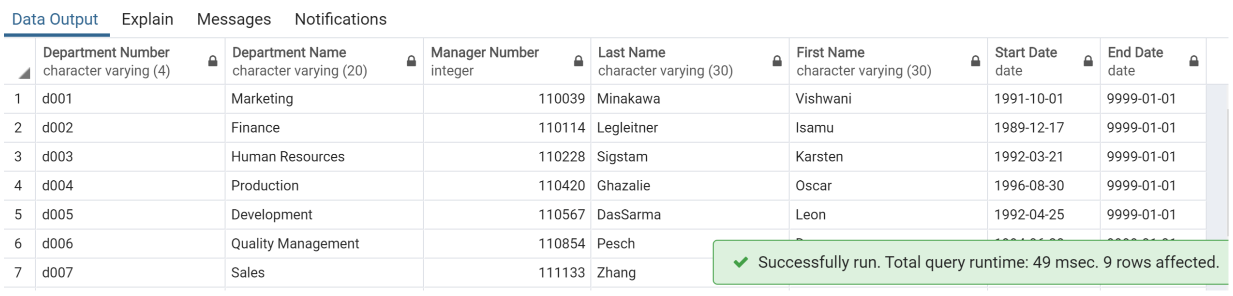

Data Output Explain Messages Notifications

	Department Number character varying (4)	Department Name character varying (20)	Manager Number integer	Last Name character varying (30)	First Name character varying (30)	Start Date date	End Date date
1	d001	Marketing	110022	Markovitch	Margareta	1985-01-01	1991-10-01
2	d001	Marketing	110039	Minakawa	Vishwani	1991-10-01	9999-01-01
3	d002	Finance	110085	Alpin	Ebru	1985-01-01	1989-12-17
4	d002	Finance	110114	Legleitner	Isamu	1989-12-17	9999-01-01
5	d003	Human Resources	110183	Ossenbruggen	Shirish	1985-01-01	1992-03-21
6	d003	Human Resources	110228	Sigstam			
7	d004	Production	110303	Wegerle			

✓ Successfully run. Total query runtime: 48 msec. 24 rows affected.

Here is just the current

```
SELECT dpt.dept_no AS "Department Number", dpt.dept_name AS "Department
 Name", dptMgr.emp_no AS "Manager Number",
    emp.last_name AS "Last Name", emp.first_name AS "First Name", dptMg
r.from_date AS "Start Date",
    dptMgr.to_date AS "End Date"
FROM "Departments" AS dpt, "Dep_Manager" AS dptMgr, "Employees" AS emp
WHERE dpt.dept_no = dptMgr.dept_no AND dptMgr.emp_no = emp.emp_no AND EX
TRACT(year FROM dptMgr.to_date)=9999;
```


Data Output Explain Messages Notifications

	Department Number character varying (4)	Department Name character varying (20)	Manager Number integer	Last Name character varying (30)	First Name character varying (30)	Start Date date	End Date date
1	d001	Marketing	110039	Minakawa	Vishwani	1991-10-01	9999-01-01
2	d002	Finance	110114	Legleitner	Isamu	1989-12-17	9999-01-01
3	d003	Human Resources	110228	Sigstam	Karsten	1992-03-21	9999-01-01
4	d004	Production	110420	Ghazalie	Oscar	1996-08-30	9999-01-01
5	d005	Development	110567	DasSarma	Leon	1992-04-25	9999-01-01
6	d006	Quality Management	110854	Pesch			
7	d007	Sales	111133	Zhang			

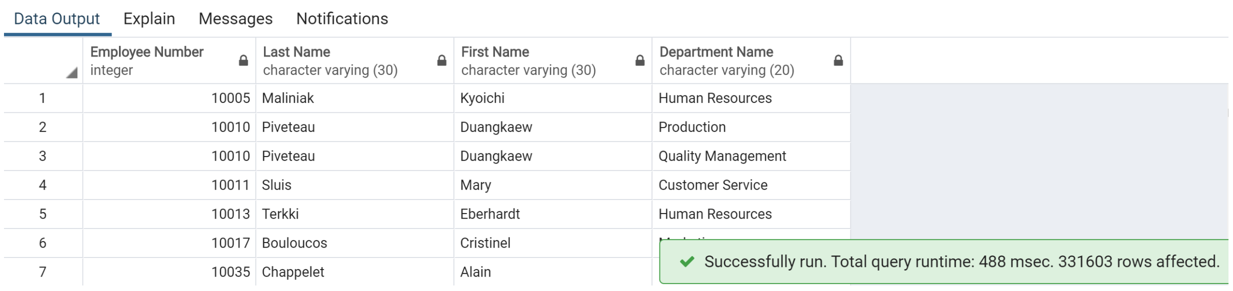
✓ Successfully run. Total query runtime: 49 msec. 9 rows affected.

4. List the department of each employee with the following information: employee number, last name, first name, and department name.

QUESTION: Is this currently or ever?

Here is the History

```
SELECT emp.emp_no AS "Employee Number", emp.last_name AS "Last Name", em
p.first_name AS "First Name",
    dpt.dept_name AS "Department Name"
FROM "Departments" AS dpt, "Dep_Emp" AS dptEmp, "Employees" AS emp
WHERE dpt.dept_no = dptEmp.dept_no AND dptEmp.emp_no = emp.emp_no;
```



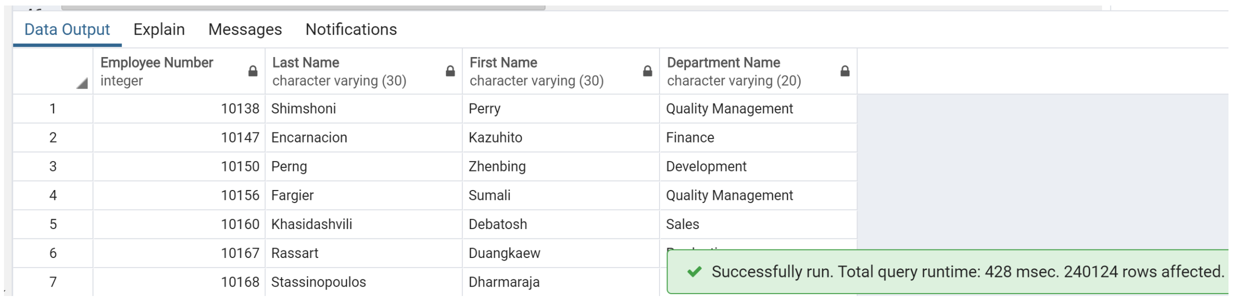
Data Output Explain Messages Notifications

	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Department Name character varying (20)
1	10005	Maliniak	Kyoichi	Human Resources
2	10010	Piveteau	Duangkaew	Production
3	10010	Piveteau	Duangkaew	Quality Management
4	10011	Sluis	Mary	Customer Service
5	10013	Terkki	Eberhardt	Human Resources
6	10017	Bouloucos	Cristinel	
7	10035	Chappelet	Alain	

✓ Successfully run. Total query runtime: 488 msec. 331603 rows affected.

Here is current

```
SELECT emp.emp_no AS "Employee Number", emp.last_name AS "Last Name", em
p.first_name AS "First Name",
    dpt.dept_name AS "Department Name"
FROM "Departments" AS dpt, "Dep_Emp" AS dptEmp, "Employees" AS emp
WHERE dpt.dept_no = dptEmp.dept_no AND dptEmp.emp_no = emp.emp_no  AND E
XTRACT(year FROM dptEmp.to_date)=9999;
```



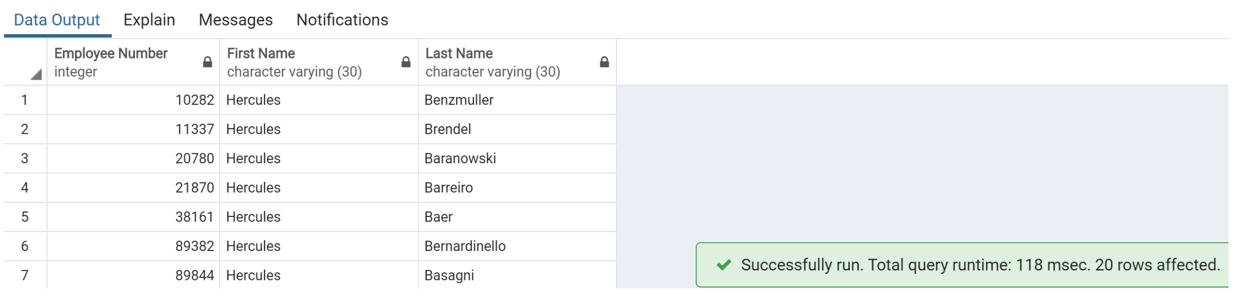
Data Output Explain Messages Notifications

	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Department Name character varying (20)
1	10138	Shimshoni	Perry	Quality Management
2	10147	Encarnacion	Kazuhito	Finance
3	10150	Perng	Zhenbing	Development
4	10156	Fargier	Sumali	Quality Management
5	10160	Khasidashvili	Debatosh	Sales
6	10167	Rassart	Duangkaew	
7	10168	Stassinopoulos	Dharmaraja	

✓ Successfully run. Total query runtime: 428 msec. 240124 rows affected.

5. List all employees whose first name is "Hercules" and last names begin with "B."

```
SELECT emp_no AS "Employee Number", first_name AS "First Name", last_nam
e AS "Last Name"
FROM "Employees"
WHERE first_name = 'Hercules' and last_name LIKE 'B%';
```



Data Output Explain Messages Notifications

	Employee Number integer	First Name character varying (30)	Last Name character varying (30)
1	10282	Hercules	Benzmuller
2	11337	Hercules	Brendel
3	20780	Hercules	Baranowski
4	21870	Hercules	Barreiro
5	38161	Hercules	Baer
6	89382	Hercules	Bernardinello
7	89844	Hercules	Basagni

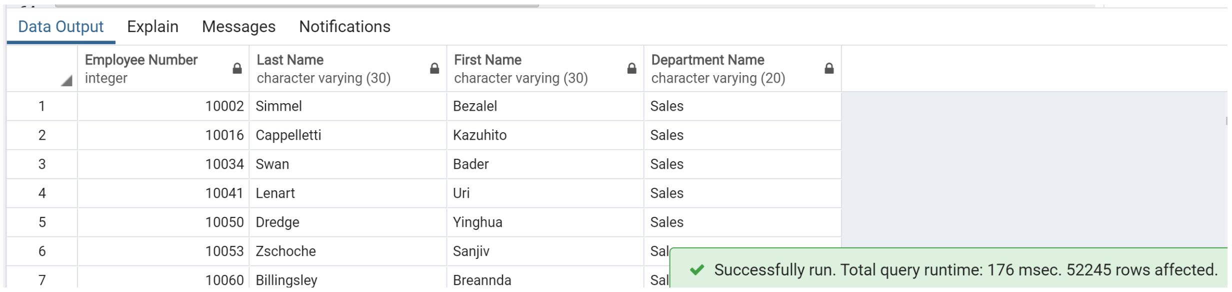
✓ Successfully run. Total query runtime: 118 msec. 20 rows affected.

6. List all employees in the Sales department, including their employee number, last name, first name, and department name.

QUESTION: Is this currently or ever?

Here is the History

```
SELECT emp.emp_no AS "Employee Number", emp.last_name AS "Last Name", em
p.first_name AS "First Name",
    dpt.dept_name AS "Department Name"
FROM "Employees" AS emp, "Departments" AS dpt, "Dep_Emp" AS dptEmp
WHERE dpt.dept_no = dptEmp.dept_no AND dptEmp.emp_no = emp.emp_no AND dp
t.dept_name = 'Sales';
```

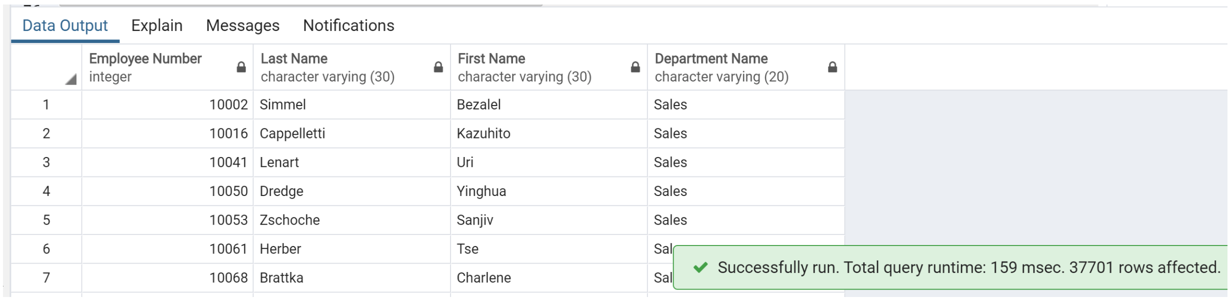


	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Department Name character varying (20)
1	10002	Simmel	Bezalel	Sales
2	10016	Cappelletti	Kazuhito	Sales
3	10034	Swan	Bader	Sales
4	10041	Lenart	Uri	Sales
5	10050	Dredge	Yinghua	Sales
6	10053	Zschoche	Sanjiv	Sal
7	10060	Billingsley	Breannda	Sal

✓ Successfully run. Total query runtime: 176 msec. 52245 rows affected.

Here is the current

```
SELECT emp.emp_no AS "Employee Number", emp.last_name AS "Last Name", em
p.first_name AS "First Name",
    dpt.dept_name AS "Department Name"
FROM "Employees" AS emp, "Departments" AS dpt, "Dep_Emp" AS dptEmp
WHERE dpt.dept_no = dptEmp.dept_no AND dptEmp.emp_no = emp.emp_no AND dp
t.dept_name = 'Sales' AND EXTRACT(year FROM dptEmp.to_date)=9999;
```



	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Department Name character varying (20)
1	10002	Simmel	Bezalel	Sales
2	10016	Cappelletti	Kazuhito	Sales
3	10041	Lenart	Uri	Sales
4	10050	Dredge	Yinghua	Sales
5	10053	Zschoche	Sanjiv	Sales
6	10061	Herber	Tse	Sal
7	10068	Brattka	Charlene	Sal

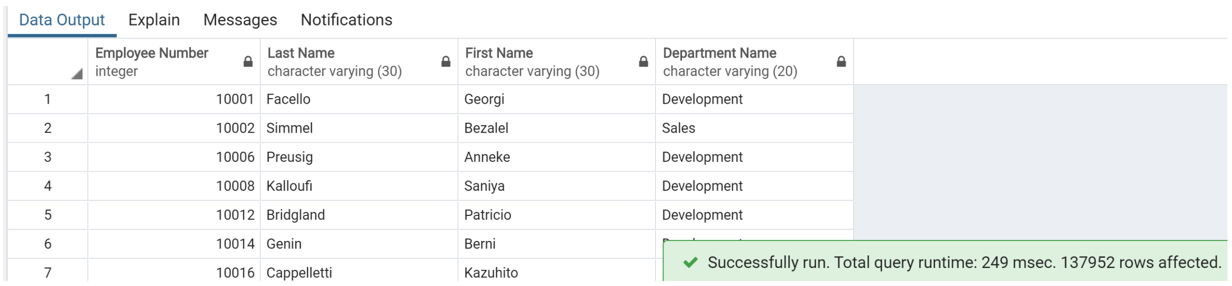
✓ Successfully run. Total query runtime: 159 msec. 37701 rows affected.

7. List all employees in the Sales and Development departments, including their employee number, last name, first name, and department name

QUESTION: Is this currently or ever?

Here is the History

```
SELECT emp.emp_no AS "Employee Number", emp.last_name AS "Last Name", em
p.first_name AS "First Name",
    dpt.dept_name AS "Department Name"
FROM "Employees" AS emp, "Departments" AS dpt, "Dep_Emp" AS dptEmp
WHERE dpt.dept_no = dptEmp.dept_no AND dptEmp.emp_no = emp.emp_no AND (d
pt.dept_name = 'Sales' OR dpt.dept_name = 'Development');
```



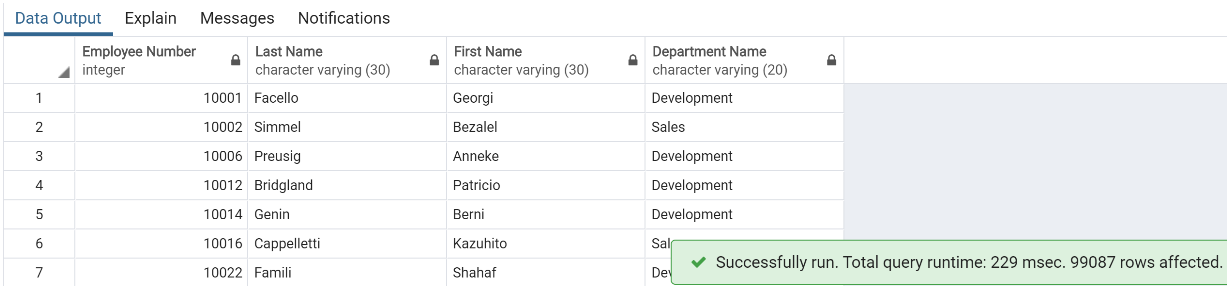
Data Output Explain Messages Notifications

	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Department Name character varying (20)
1	10001	Facello	Georgi	Development
2	10002	Simmel	Bezalel	Sales
3	10006	Preusig	Anneke	Development
4	10008	Kalloufi	Saniya	Development
5	10012	Bridgland	Patricio	Development
6	10014	Genin	Berni	[illegible]
7	10016	Cappelletti	Kazuhito	

✓ Successfully run. Total query runtime: 249 msec. 137952 rows affected.

Here is the current

```
SELECT emp.emp_no AS "Employee Number", emp.last_name AS "Last Name", em
p.first_name AS "First Name",
    dpt.dept_name AS "Department Name"
FROM "Employees" AS emp, "Departments" AS dpt, "Dep_Emp" AS dptEmp
WHERE dpt.dept_no = dptEmp.dept_no AND dptEmp.emp_no = emp.emp_no AND (d
pt.dept_name = 'Sales' OR dpt.dept_name = 'Development') AND EXTRACT(yea
r FROM dptEmp.to_date)=9999;
```



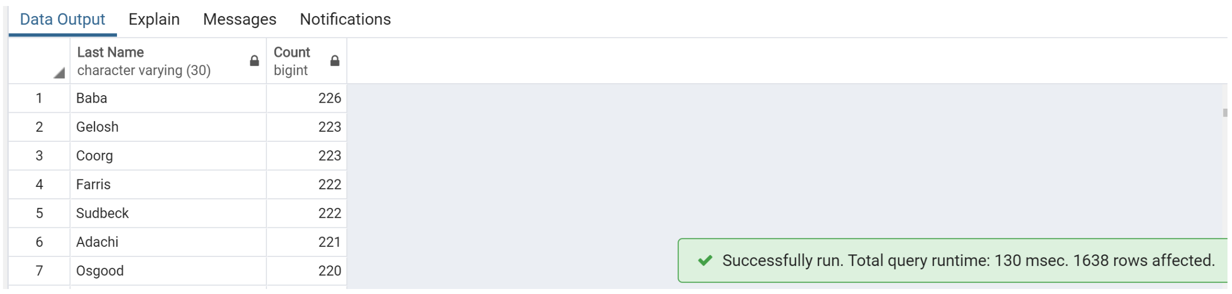
Data Output Explain Messages Notifications

	Employee Number integer	Last Name character varying (30)	First Name character varying (30)	Department Name character varying (20)
1	10001	Facello	Georgi	Development
2	10002	Simmel	Bezalel	Sales
3	10006	Preusig	Anneke	Development
4	10012	Bridgland	Patricio	Development
5	10014	Genin	Berni	Development
6	10016	Cappelletti	Kazuhito	Sal
7	10022	Famili	Shahaf	Dev

✓ Successfully run. Total query runtime: 229 msec. 99087 rows affected.

8. In descending order, list the frequency count of employee last names, i.e., how many employees share each last name.

```
SELECT last_name AS "Last Name", COUNT(last_name) AS "Count"
FROM "Employees"
GROUP BY last_name
ORDER BY "Count" DESC;
```



Data Output Explain Messages Notifications

	Last Name character varying (30)	Count bigint
1	Baba	226
2	Gelosh	223
3	Coorg	223
4	Farris	222
5	Sudbeck	222
6	Adachi	221
7	Osgood	220

✓ Successfully run. Total query runtime: 130 msec. 1638 rows affected.

Phase 2 - Graphical Analysis

Imports

Pandas

Data manipulation and analysis

MatPlotLib Pyplot

2D plotting

Datetime

Dates and time

Numpy

Supports large, multi-dimensional arrays and matrix manipulation and high level mathematical functions on these arrays

SQLAlchemy

Database Import

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import datetime
import numpy as np
from sqlalchemy import create_engine

# Import db pw
from keys import pw
```

Reusable References

In [2]:
```
BOLD = '\033[1m'
END = '\033[0m'
```

Create the Database Connection

In [3]:
```
#db_uri = 'postgresql://postgres:' + pw + '@localhost:5432/SQLChallenge'
#engine = create_engine(db_uri)
#connection = engine.connect()
```

This code will be replaced with accessing the database

In [4]:
```
# Import Employees, Titles and Salaries csv's
emp_csv = './data/employees.csv'
empDateCols = ['birth_date', 'hire_date']
employees = pd.read_csv(emp_csv, parse_dates=empDateCols)

title_csv = './data/titles.csv'
titleDateCols = ['from_date', 'to_date']
titles = pd.read_csv(title_csv, parse_dates=titleDateCols)

sal_csv = './data/salaries.csv'
salaryDateCols = ['from_date', 'to_date']
salaries = pd.read_csv(sal_csv, parse_dates=salaryDateCols)
```

Create a bar chart of average salary by title

Assumption

- Only include current employee salaries since historic salaries do not represent current day value

Start by selecting only current employees

In [5]:

```
# Start with the salaries table and select only rows that contain a to_date y
# First, verify the data types of the to_date column
print(BOLD + 'The to_date column of the salary dataframe is of type:' + END)
print(type(salaries['to_date'].iat[0]))

# Alternatively we can look at all columns at one time
print(BOLD + '\n\nThe data types in the salary dataframe are:' + END)
print(salaries.dtypes)

# Now lets check value_counts
print(BOLD + '\n\nThe value counts for the to_date column are:' + END)
print(salaries['to_date'].value_counts().sort_values(ascending=False))

print(BOLD + 'NOTE: There are no values for the year 9999, so we cannot use t
# We can confirm, however that there are no values by creating a dataframe as
current_salary = salaries.loc[salaries['to_date'].dt.year==9999,:]
print(BOLD + '\n\nThe results of searching for to_date values in the year 999
print(current_salary)

print(BOLD + 'NOTE:  The dataframe is empty which aligns with our expectation
```

```
The to_date column of the salary dataframe is of type:
<class 'pandas._libs.tslibs.timestamps.Timestamp'>


The data types in the salary dataframe are:
emp_no                int64
salary                int64
from_date    datetime64[ns]
to_date      datetime64[ns]
dtype: object


The value counts for the to_date column are:
1995-10-29    120
1997-10-26    117
1989-10-29    115
1990-10-28    115
1998-10-25    111
             ... 
1985-11-28      1
1985-05-23      1
1985-03-11      1
1985-09-08      1
1985-04-20      1
Name: to_date, Length: 5568, dtype: int64
NOTE: There are no values for the year 9999, so we cannot use this column t
o determine current employees.


The results of searching for to_date values in the year 9999 are as expecte
d with no returned values:
Empty DataFrame
Columns: [emp_no, salary, from_date, to_date]
Index: []
NOTE:  The dataframe is empty which aligns with our expectations
```

In [6]:

```
# Results above show that not all to_date columns in this database use the sa
# Instead, lets check the titles table

# we can look at all columns at one time to check for datatypes
print(BOLD + 'The data types in the titles dataframe are:' + END)
print(titles.dtypes)
print(BOLD + 'NOTE:  The to_date column would not convert to datetime' + END)

# Now lets check value_counts
print(BOLD + '\n\nThe value counts for the to_date column are:' + END)
print(titles['to_date'].value_counts(ascending=True))
print(BOLD + 'NOTE:  There are roughly 24k employees currently.' + END)

# NOTE - The titles table to_date column will not come in as a date, therefor
print(BOLD + "\n\nTitles dataframe filtered for current employees (to_date =
titles['to_date'].value_counts(ascending=False)
current_title = titles.loc[titles['to_date'] == '9999-01-01',:]
current_title.head()
```

```
The data types in the titles dataframe are:
emp_no                int64
title                object
from_date    datetime64[ns]
to_date              object
dtype: object
NOTE:  The to_date column would not convert to datetime


The value counts for the to_date column are:
1988-10-27         1
1986-02-01         1
1988-03-30         1
1988-08-26         1
1985-08-07         1
               ...
2001-06-26        79
2000-08-15        81
1997-10-26        88
1998-10-25        91
9999-01-01    240124
Name: to_date, Length: 5888, dtype: int64
NOTE:  There are roughly 24k employees currently.


Titles dataframe filtered for current employees (to_date = 9999-01-01):
```

Out[6]:

	emp_no	title	from_date	to_date
0	10001	Senior Engineer	1986-06-26	9999-01-01
1	10002	Staff	1996-08-03	9999-01-01
2	10003	Senior Engineer	1995-12-03	9999-01-01
4	10004	Senior Engineer	1995-12-01	9999-01-01
5	10005	Senior Staff	1996-09-12	9999-01-01

Now, merge the titles (current employees), employees and salaries dataframes

In [7]:
```
# Start with merging employees to current titles
print(BOLD + 'Employees merged with Titles (current employees only)' + END)
emp_title = employees.merge(current_title, on='emp_no')
emp_title.head()
```

```
Employees merged with Titles (current employees only)
```

Out[7]:

	emp_no	birth_date	first_name	last_name	gender	hire_date	title	from_date	to_date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26	Senior Engineer	1986-06-26	9999-01-01
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	Staff	1996-08-03	9999-01-01
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28	Senior Engineer	1995-12-03	9999-01-01
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	Senior Engineer	1995-12-01	9999-01-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12	Senior Staff	1996-09-12	9999-01-01

In [8]:
```
# Now merge the salaries
print(BOLD + 'Salaries merged with Employees and Titles (current employees or
emp_title_salary = emp_title.merge(salaries, on='emp_no').sort_values(['title

# Finally, let's select only the columns we need
emp_title_salary = emp_title_salary[['title', 'salary']]
emp_title_salary.head()
```

```
Salaries merged with Employees and Titles (current employees only)
```

Out[8]:

	title	salary
0	Assistant Engineer	41396
1	Assistant Engineer	66958
2	Assistant Engineer	40000
3	Assistant Engineer	55072
4	Assistant Engineer	40000

Next, groupby title and create statistics used for graphing

In [9]:

```
# groupby title and create relevant statistics on the salary
print(BOLD + 'Salary statistics by Title' + END)
statistics_salary_by_title = emp_title_salary.groupby(['title']).aggregate(
    {
        'salary':['mean', 'median', 'min', 'max', 'count', 'sem']
    }
).sort_values(by=['title'], ascending=True)
statistics_salary_by_title.columns = statistics_salary_by_title.columns.dropl
statistics_salary_by_title.sort_values(['title'], ascending=True).reset_index
```

```
Salary statistics by Title
```

Out[9]:

	title	mean	median	min	max	count	sem
0	Assistant Engineer	48436.856187	44436.5	40000	99683	3588	170.178787
1	Engineer	48532.428751	44489.0	40000	100683	30983	58.671993
2	Manager	49600.555556	45169.0	40000	71148	9	4207.742157
3	Senior Engineer	48501.994322	44486.0	40000	110449	85939	34.983389
4	Senior Staff	58511.960170	56450.0	40000	129492	82024	57.601367
5	Staff	58448.920003	56387.0	40000	127041	25526	103.146245
6	Technique Leader	48532.833762	44427.0	40000	104065	12055	93.750308

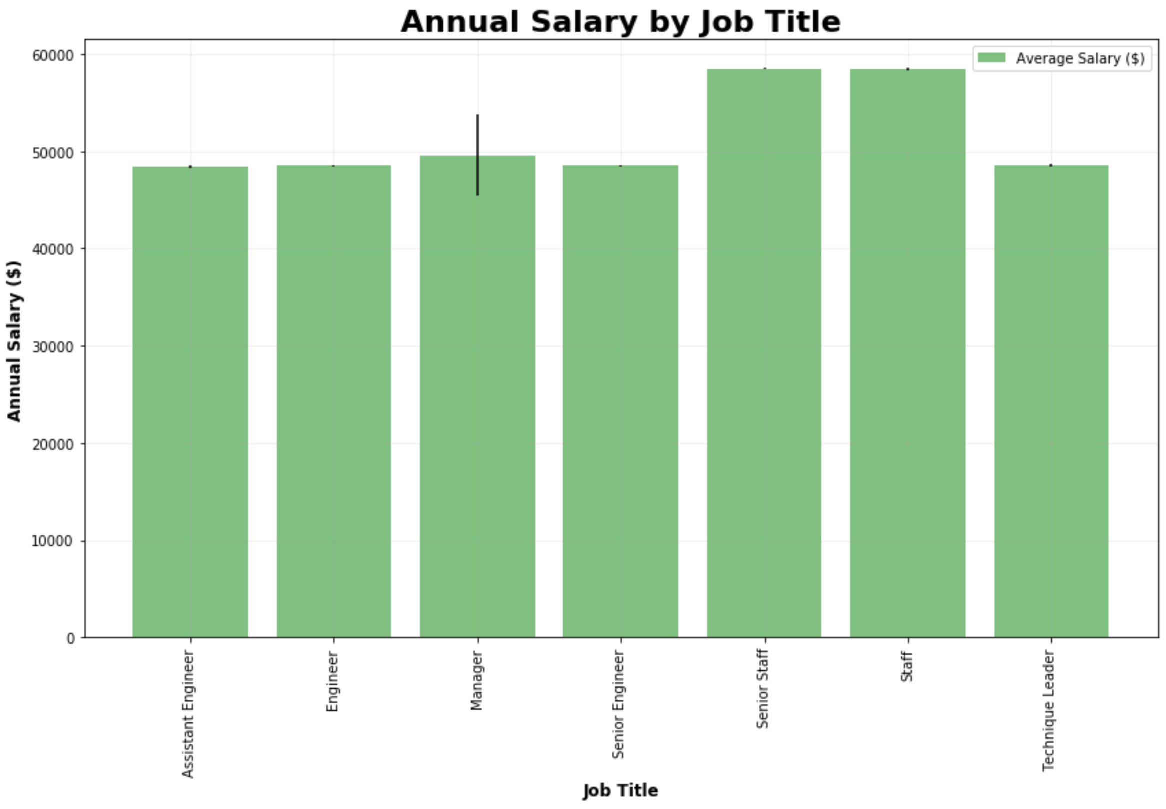
Create a Bar Chart to Show Average Salary by Job Title (and include error bars)

In [10]:
```
# Create plot
fig, ax = plt.subplots(figsize=(14,8))
plt.bar(statistics_salary_by_title.index, statistics_salary_by_title['mean'],
        alpha=0.5, yerr=statistics_salary_by_title['sem'], label='Average Sal

# Add labels and formatting
plt.xlabel("Job Title", weight='bold', size=12)
plt.ylabel("Annual Salary ($)", weight='bold', size=12)
plt.title("Annual Salary by Job Title", weight='bold', size=22)
plt.xticks(statistics_salary_by_title.index, rotation=90)
plt.legend(loc='best')
plt.grid(alpha=0.2)

# Save the chart out
plt.savefig("./Output/SalaryByTitleBar.png")

# Show plot
plt.show()
```



Interesting... why does Manager have such large error bars?

Let's dig a little more and plot the distributions via a box plot

First, create a function

In [11]:

```
# Function comparing populations by Boxplots
def boxPlotCompare(srStaff, staff, mgr, techLdr, eng, srEng, asstEng, title):

    # Set the figure size
    fig = plt.figure(figsize=(14,8))
    axBox = fig.add_subplot()

    # Show box plots of the data
    box_plot_data=[srStaff, staff, mgr, techLdr, eng, srEng, asstEng]
    plt.boxplot(box_plot_data)

    # Format the chart
    plt.title(title, color='k', size=24, weight='bold')
    plt.xticks([1, 2, 3, 4, 5, 6, 7], ['Assistant Engineer', 'Engineer', 'Mar
    plt.xlabel("Job Title", size=14, weight='bold')
    plt.ylabel("Salary ($)", size=14, weight='bold')

    # Save the chart out
    plt.savefig("./Output/SalaryByTitleBoxPlot.png")

    # Show the chart
    plt.show()

    return
```

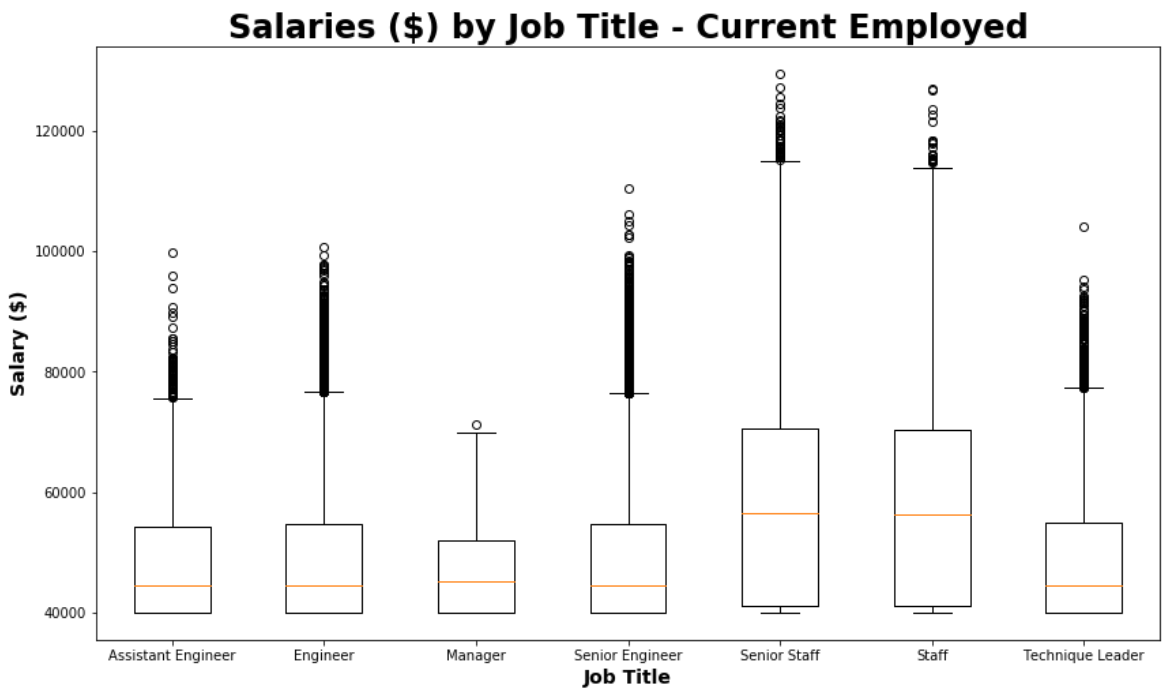
Next, create a series for each job title

In [12]:

```
# Create a salary series for each title
srStaff = emp_title_salary.loc[(emp_title_salary['title']=='Senior Staff'),'s
staff = emp_title_salary.loc[(emp_title_salary['title']=='Staff'),'salary']
mgr = emp_title_salary.loc[(emp_title_salary['title']=='Manager'),'salary']
techLdr = emp_title_salary.loc[(emp_title_salary['title']=='Technique Leader'
eng = emp_title_salary.loc[(emp_title_salary['title']=='Engineer'),'salary']
srEng = emp_title_salary.loc[(emp_title_salary['title']=='Senior Engineer'),'
asstEng = emp_title_salary.loc[(emp_title_salary['title']=='Assistant Enginee
```

Finally, plot the box plots

In [13]:
```
# Plot salaries by title in box plots
boxPlotCompare(asstEng, eng, mgr, srEng, srStaff, staff, techLdr, "Salaries (
```



Manager has a small range (25 percentile to 75 percentile) and the fewest outliers, but the largest errors.

Let's dig more...

This time, let's look at a scatter plot of the individual data and plot the median as a reference

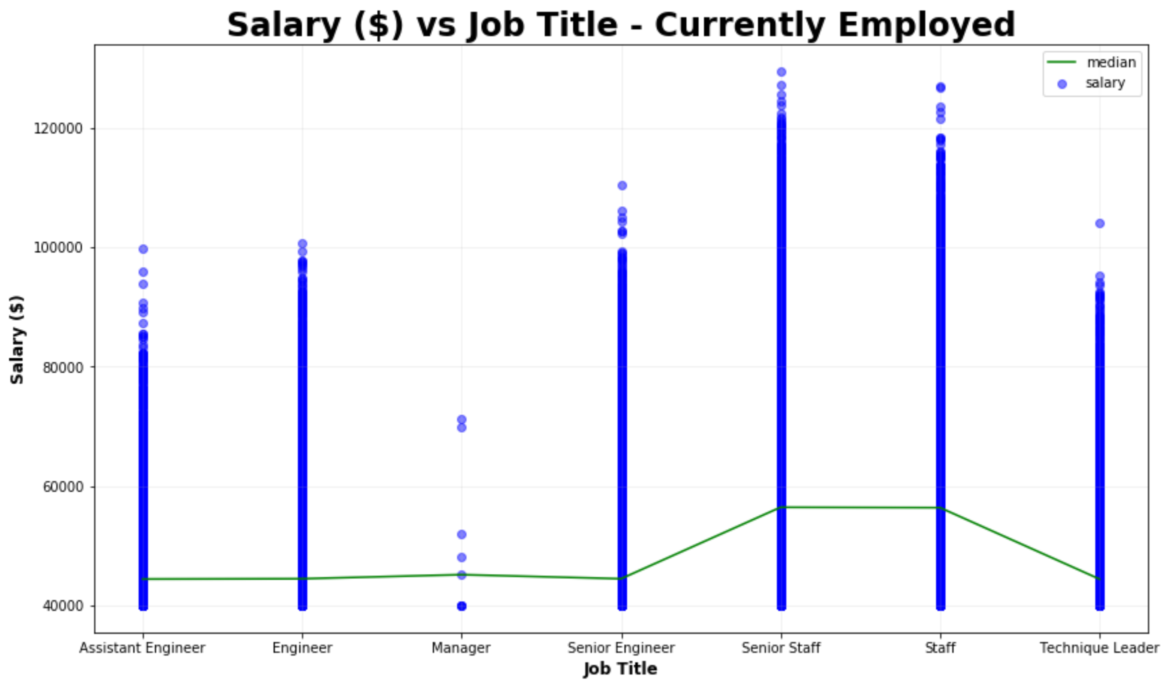
In [14]:

```
# Plot the data
fig = plt.figure(figsize=(14,8))
ax = fig.add_subplot()
plt.scatter(emp_title_salary["title"], emp_title_salary["salary"], c="b", alp
plt.plot(statistics_salary_by_title.index, statistics_salary_by_title['mediar

# format the chart
plt.title(f'Salary ($) vs Job Title - Currently Employed', color='k', size=24
plt.xlabel('Job Title', weight='bold', size=12)
plt.ylabel('Salary ($)', weight='bold', size=12)
plt.grid(alpha=0.2)
plt.legend(loc='best')

# Save the chart out
plt.savefig("./Output/SalaryByTitleScatterAndMedian.png")

# Show the plot
plt.show()
```



Interesting, there is very little data for Manager so there is a higher spread between points

Let's now check how many data points there are

In [15]:
```
# Redisplay the statistics table
print(BOLD + 'Statistics of Salaries by Job Title' + END)
print(statistics_salary_by_title)

print(BOLD + 'NOTE:  Manager only has 9 data points compared to thousands for
```

```
Statistics of Salaries by Job Title
                           mean   median    min     max  count          se
m
title
Assistant Engineer  48436.856187  44436.5  40000   99683   3588   170.17878
7
Engineer            48532.428751  44489.0  40000  100683  30983    58.67199
3
Manager             49600.555556  45169.0  40000   71148      9  4207.74215
7
Senior Engineer     48501.994322  44486.0  40000  110449  85939    34.98338
9
Senior Staff        58511.960170  56450.0  40000  129492  82024    57.60136
7
Staff               58448.920003  56387.0  40000  127041  25526   103.14624
5
Technique Leader    48532.833762  44427.0  40000  104065  12055    93.75030
8
NOTE:  Manager only has 9 data points compared to thousands for the other t
itles.
```

Now, per the last item requested by the boss, look up employee id

Look up Employee ID = 499942

In [16]:
```
my_info = employees.loc[employees["emp_no"] == 499942, :]
my_info
```

Out[16]:

	emp_no	birth_date	first_name	last_name	gender	hire_date
299966	499942	1963-01-10	April	Foolsday	F	1997-02-10

NOTE the fact that suspicians were correct and the data is fake.