

SQL & Python



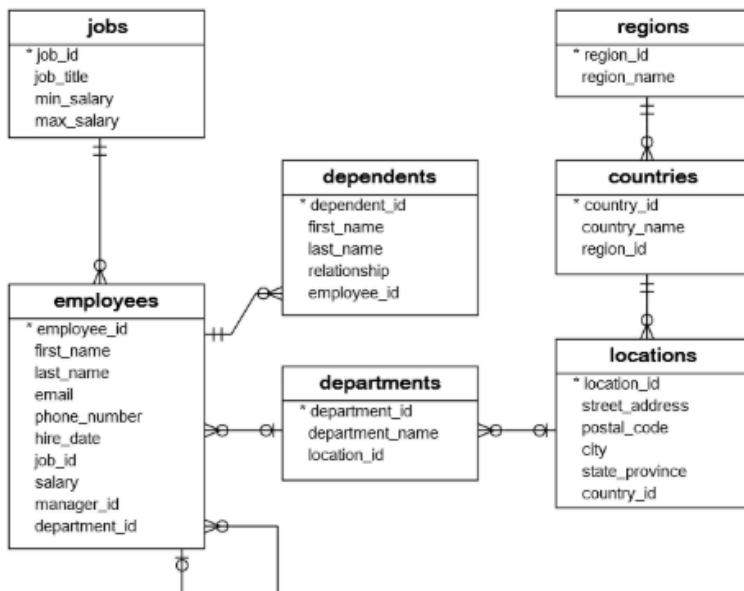
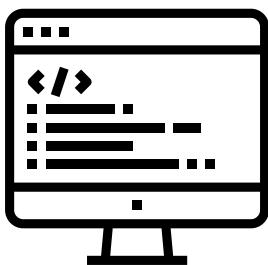
| | |
|---------------|--|
| Project Title | Exploratory Analysis of Compensation Trends in Sample HR Dataset |
|---------------|--|

| | |
|--------------|-------------|
| Project Date | Summer 2025 |
|--------------|-------------|

Project Overview

Sample SQLite Data was used from the following source: sqltutorial.org/sql-sample-database/. The sample data contains seven tables: jobs, employees, dependents, departments, regions, countries, and locations. The sample data was analyzed in Jupyter Notebook, using descriptive and inferential statistical methods. Python modules utilized were pandas, matplotlib, statsmodel, and sqlite3.

Data Schema



Key Highlights/Deliverables

- Created a normalized employee database with department and salary info
- Wrote complex SQL queries including joins, CTEs, and aggregations
- Built salary quartile classifications and analyzed tenure vs. pay
- Visualized pay disparities using bar charts and scatter plots
- Interpreted regression results to model salary predictors

```
In [22]: #Import Needed Modules
import sqlite3
import pandas as pd

# Create a connection to the database, result is a Connection object
conn = sqlite3.connect("compensation1.db")
# Create a Cursor object that allows user to execute SQL commands and fetch results from the database
cursor = conn.cursor()

# Creates tables with designated columns
#Foreign keys are designated and referenced to table that holds its primary key for database integrity
cursor.executescript("""
CREATE TABLE IF NOT EXISTS regions (
    region_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    region_name text NOT NULL
);

CREATE TABLE IF NOT EXISTS countries (
    country_id text NOT NULL,
    country_name text NOT NULL,
    region_id INTEGER NOT NULL,
    PRIMARY KEY (country_id ASC),
    FOREIGN KEY (region_id) REFERENCES regions (region_id) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS locations (
    location_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    street_address text,
    postal_code text,
    city text NOT NULL,
    state_province text,
    country_id INTEGER NOT NULL,
    FOREIGN KEY (country_id) REFERENCES countries (country_id) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS departments (
    department_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    department_name text NOT NULL,
    location_id INTEGER NOT NULL,
    FOREIGN KEY (location_id) REFERENCES locations (location_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
);

CREATE TABLE IF NOT EXISTS jobs (
    job_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    job_title text NOT NULL,
    min_salary double NOT NULL,
    max_salary double NOT NULL
);

CREATE TABLE IF NOT EXISTS employees (
    employee_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    first_name text,
    last_name text NOT NULL,
    email text NOT NULL,
    phone_number text,
    hire_date text NOT NULL,
    job_id INTEGER NOT NULL,
    salary double NOT NULL,
    manager_id INTEGER,
    department_id INTEGER NOT NULL,
    FOREIGN KEY (job_id) REFERENCES jobs (job_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (department_id) REFERENCES departments (department_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (manager_id) REFERENCES employees (employee_id) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS dependents (
    dependent_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    first_name text NOT NULL,
    last_name text NOT NULL,
    relationship text NOT NULL,
    employee_id INTEGER NOT NULL,
    FOREIGN KEY (employee_id) REFERENCES employees (employee_id) ON DELETE CASCADE ON UPDATE CASCADE
);
"""

)
```

Out[22]: <sqlite3.Cursor at 0x296942c7540>

```
In [27]: #Inserts values into each table created using INSERT INTO table_name(columns) VALUES (values)
cursor.executescript("""
INSERT OR IGNORE INTO regions(region_id,region_name) VALUES (1,'Europe');
INSERT OR IGNORE INTO regions(region_id,region_name) VALUES (2,'Americas');
INSERT OR IGNORE INTO regions(region_id,region_name) VALUES (3,'Asia');
```

```
INSERT OR IGNORE INTO regions(region_id,region_name) VALUES (4,'Middle East and Africa');

INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('AR','Argentina',2);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('AU','Australia',3);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('BE','Belgium',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('BR','Brazil',2);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('CA','Canada',2);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('CH','Switzerland',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('CN','China',3);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('DE','Germany',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('DK','Denmark',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('EG','Egypt',4);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('FR','France',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('HK','HongKong',3);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('IL','Israel',4);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('IN','India',3);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('IT','Italy',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('JP','Japan',3);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('KW','Kuwait',4);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('MX','Mexico',2);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('NG','Nigeria',4);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('NL','Netherlands',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('SG','Singapore',3);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('UK','United Kingdom',1);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('US','United States of America',2);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('ZM','Zambia',4);
INSERT OR IGNORE INTO countries(country_id,country_name,region_id) VALUES ('ZW','Zimbabwe',4);

INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (1400,';
INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (1500,';
INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (1700,';
INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (1800,';
INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (2400,';
INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (2500,';
INSERT OR IGNORE INTO locations(location_id,street_address,postal_code,city,state_province,country_id) VALUES (2700,';

INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (1,'Public Accountant',4200.00,9000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (2,'Accounting Manager',8200.00,16000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (3,'Administration Assistant',3000.00,6000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (4,'President',20000.00,40000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (5,'Administration Vice President',15000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (6,'Accountant',4200.00,9000.00);
```

```

INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (7,'Finance Manager',8200.00,16000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (8,'Human Resources Representative',4000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (9,'Programmer',4000.00,10000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (10,'Marketing Manager',9000.00,15000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (11,'Marketing Representative',4000.00,9000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (12,'Public Relations Representative',4500.00,12000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (13,'Purchasing Clerk',2500.00,5500.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (14,'Purchasing Manager',8000.00,15000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (15,'Sales Manager',10000.00,20000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (16,'Sales Representative',6000.00,12000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (17,'Shipping Clerk',2500.00,5500.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (18,'Stock Clerk',2000.00,5000.00);
INSERT OR IGNORE INTO jobs(job_id,job_title,min_salary,max_salary) VALUES (19,'Stock Manager',5500.00,8500.00);

INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (1,'Administration',1700);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (2,'Marketing',1800);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (3,'Purchasing',1700);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (4,'Human Resources',2400);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (5,'Shipping',1500);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (6,'IT',1400);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (7,'Public Relations',2700);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (8,'Sales',2500);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (9,'Executive',1700);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (10,'Finance',1700);
INSERT OR IGNORE INTO departments(department_id,department_name,location_id) VALUES (11,'Accounting',1700);

""")

```

Out[27]: <sqlite3.Cursor at 0x296942c7540>

In [3]: #First, we will explore the tables within the database using `sqlite_master`, the special internal table
#for SQLite databases

```

tables_df = pd.read_sql("SELECT name FROM sqlite_master WHERE type='table';", conn)

#Outputs the names of all tables, as well as sqlite_sequence another special internal table that tracks the
#Last value for AUTOINCREMENT
print(tables_df)

```

```
      name
0      regions
1  sqlite_sequence
2    countries
3   locations
4  departments
5     jobs
6   employees
7  dependents
```

```
In [4]: #Next, lets take a look at the employees table
employees_table= pd.read_sql("SELECT * FROM employees LIMIT 5;", conn)
print(employees_table)
```

```
employee_id first_name last_name          email \
0           100    Steven     King  steven.king@sqltutorial.org
1           101    Neena    Kochhar neena.kochhar@sqltutorial.org
2           102      Lex  De Haan  lex.de haan@sqltutorial.org
3           103  Alexander   Hunold alexander.hunold@sqltutorial.org
4           104    Bruce    Ernst  bruce.ernst@sqltutorial.org

phone_number    hire_date    job_id    salary  manager_id  department_id
0  515.123.4567  1987-06-17      4  24000.0        NaN            9
1  515.123.4568  1989-09-21      5  17000.0       100.0            9
2  515.123.4569  1993-01-13      5  17000.0       100.0            9
3  590.423.4567  1990-01-03      9   9000.0       102.0            6
4  590.423.4568  1991-05-21      9   6000.0       103.0            6
```

```
In [5]: #Let's import matplotlib so we can graph some of our inquiries later
import matplotlib.pyplot as plt

#Let's begin by asking questions surrounding the employee table to better understand this organization

#How many employees are there?
num_employee = pd.read_sql("SELECT COUNT(employee_id) AS 'Number of Employees' FROM employees;", conn)
print(num_employee)
```

```
Number of Employees
0                40
```

```
In [6]: #How many employees are in each department?
q_num_dept = """
WITH temp_table AS (
```

```

        SELECT department_id,
               COUNT(*) AS num_emp
        FROM employees
    GROUP BY department_id
    )
SELECT d.department_name as 'Department',
       num_emp as 'Number of Employees'
  FROM departments d
 JOIN temp_table
    ON d.department_id = temp_table.department_id
 ORDER BY d.department_name;
"""

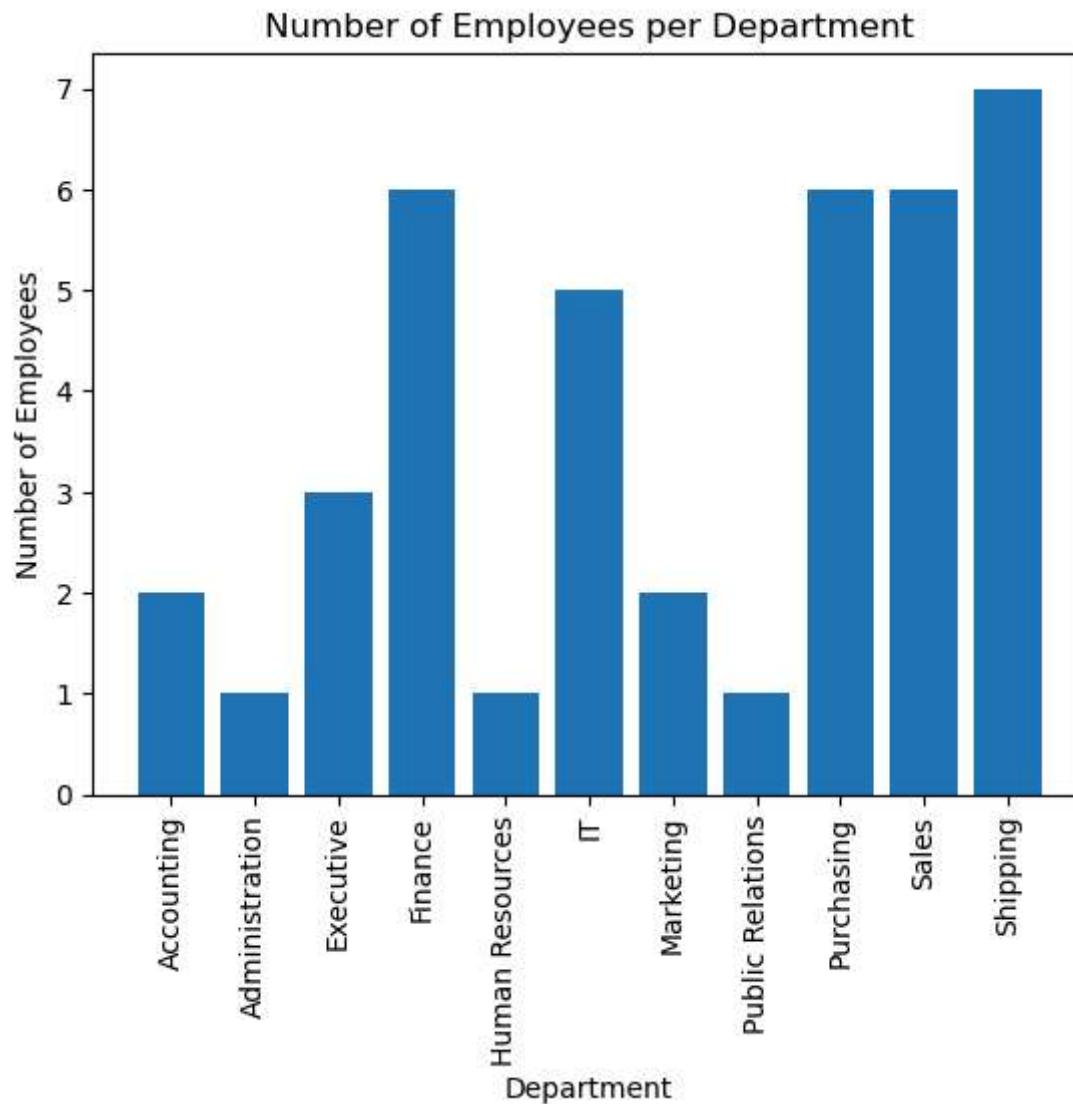
num_employee = pd.read_sql(q_num_dept, conn)
print(num_employee, end = "\n\n")

```

| | Department | Number of Employees |
|----|------------------|---------------------|
| 0 | Accounting | 2 |
| 1 | Administration | 1 |
| 2 | Executive | 3 |
| 3 | Finance | 6 |
| 4 | Human Resources | 1 |
| 5 | IT | 5 |
| 6 | Marketing | 2 |
| 7 | Public Relations | 1 |
| 8 | Purchasing | 6 |
| 9 | Sales | 6 |
| 10 | Shipping | 7 |

In [7]:

```
#Let's plot this into a bar chart for a visual representation
plt.bar(num_employee['Department'], num_employee['Number of Employees'])
plt.xticks(rotation=90)
plt.title('Number of Employees per Department')
plt.xlabel('Department')
plt.ylabel('Number of Employees')
plt.show()
```



```
In [8]: #What is the lowest and highest salary in each department?  
q_low_high = """  
    WITH temp_table AS (  
        SELECT department_id,  
            MIN(salary) as low_salary,  
            MAX(salary) as high_salary  
        FROM employees  
        GROUP BY department_id
```

```

        )
SELECT d.department_name as 'Department',
       low_salary as 'Lowest Salary',
       high_salary as 'Highest Salary'
  FROM departments d
 JOIN temp_table
    ON d.department_id = temp_table.department_id
 ORDER BY department_name;
"""

low_high = pd.read_sql(q_low_high, conn)
print(low_high, end = "\n\n")

```

| | Department | Lowest Salary | Highest Salary |
|----|------------------|---------------|----------------|
| 0 | Accounting | 8300.0 | 12000.0 |
| 1 | Administration | 4400.0 | 4400.0 |
| 2 | Executive | 17000.0 | 24000.0 |
| 3 | Finance | 6900.0 | 12000.0 |
| 4 | Human Resources | 6500.0 | 6500.0 |
| 5 | IT | 4200.0 | 9000.0 |
| 6 | Marketing | 6000.0 | 13000.0 |
| 7 | Public Relations | 10000.0 | 10000.0 |
| 8 | Purchasing | 2500.0 | 11000.0 |
| 9 | Sales | 6200.0 | 14000.0 |
| 10 | Shipping | 2700.0 | 8200.0 |

In [10]: #For each employee's salary, let's calculate which quartile they are in based on their job's salary range
#For example, if an employee makes \$25,000 and the salary range is \$21,000 to \$32,000, then they are in Quartile 1.

#First, we gather the table that shows the name of the employee, their salary, their position, and the salary range
#for their position.

```

q_quartiles = """
WITH temp_table AS (
    SELECT e.first_name,
           e.last_name,
           e.salary,
           j.min_salary,
           j.max_salary
      FROM employees e
     JOIN jobs j
        ON e.job_id = j.job_id
)

```

```
SELECT * FROM temp_table;  
***  
quartiles = pd.read_sql(q_quartiles,conn)  
print(quartiles)
```

| | first_name | last_name | salary | min_salary | max_salary | |
|----|------------|-------------|----------|------------|------------|---------|
| 0 | Steven | King | 24000.0 | 20000.0 | 40000.0 | |
| 1 | Neena | Kochhar | 17000.0 | 15000.0 | 30000.0 | |
| 2 | Lex | De Haan | 17000.0 | 15000.0 | 30000.0 | |
| 3 | Alexander | Hunold | 9000.0 | 4000.0 | 10000.0 | |
| 4 | Bruce | Ernst | 6000.0 | 4000.0 | 10000.0 | |
| 5 | David | Austin | 4800.0 | 4000.0 | 10000.0 | |
| 6 | Valli | Pataballa | 4800.0 | 4000.0 | 10000.0 | |
| 7 | Diana | Lorentz | 4200.0 | 4000.0 | 10000.0 | |
| 8 | Nancy | Greenberg | 12000.0 | 8200.0 | 16000.0 | |
| 9 | Daniel | Faviet | 9000.0 | 4200.0 | 9000.0 | |
| 10 | John | Chen | 8200.0 | 4200.0 | 9000.0 | |
| 11 | Ismael | Sciarra | 7700.0 | 4200.0 | 9000.0 | |
| 12 | Jose | Manuel | Urman | 7800.0 | 4200.0 | 9000.0 |
| 13 | Luis | | Popp | 6900.0 | 4200.0 | 9000.0 |
| 14 | | Den | Raphaely | 11000.0 | 8000.0 | 15000.0 |
| 15 | Alexander | | Khoo | 3100.0 | 2500.0 | 5500.0 |
| 16 | Shelli | | Baida | 2900.0 | 2500.0 | 5500.0 |
| 17 | Sigal | | Tobias | 2800.0 | 2500.0 | 5500.0 |
| 18 | Guy | | Himuro | 2600.0 | 2500.0 | 5500.0 |
| 19 | Karen | Colmenares | | 2500.0 | 2500.0 | 5500.0 |
| 20 | Matthew | | Weiss | 8000.0 | 5500.0 | 8500.0 |
| 21 | Adam | | Fripp | 8200.0 | 5500.0 | 8500.0 |
| 22 | Payam | | Kaufling | 7900.0 | 5500.0 | 8500.0 |
| 23 | Shanta | | Vollman | 6500.0 | 5500.0 | 8500.0 |
| 24 | Irene | Mikkilineni | | 2700.0 | 2000.0 | 5000.0 |
| 25 | John | Russell | | 14000.0 | 10000.0 | 20000.0 |
| 26 | Karen | Partners | | 13500.0 | 10000.0 | 20000.0 |
| 27 | Jonathon | | Taylor | 8600.0 | 6000.0 | 12000.0 |
| 28 | Jack | Livingston | | 8400.0 | 6000.0 | 12000.0 |
| 29 | Kimberely | | Grant | 7000.0 | 6000.0 | 12000.0 |
| 30 | Charles | Johnson | | 6200.0 | 6000.0 | 12000.0 |
| 31 | Sarah | Bell | | 4000.0 | 2500.0 | 5500.0 |
| 32 | Britney | Everett | | 3900.0 | 2500.0 | 5500.0 |
| 33 | Jennifer | Whalen | | 4400.0 | 3000.0 | 6000.0 |
| 34 | Michael | Hartstein | | 13000.0 | 9000.0 | 15000.0 |
| 35 | Pat | Fay | | 6000.0 | 4000.0 | 9000.0 |
| 36 | Susan | Mavris | | 6500.0 | 4000.0 | 9000.0 |
| 37 | Hermann | Baer | | 10000.0 | 4500.0 | 10500.0 |
| 38 | Shelley | Higgins | | 12000.0 | 8200.0 | 16000.0 |
| 39 | William | Gietz | | 8300.0 | 4200.0 | 9000.0 |

```
In [11]: #Since each employee has their own individual salary range, we will need to find their relative position  
#in that salary range. This will be stored in new column named "Salary Range Position"  
quartiles['Salary Range Position'] = round((quartiles['salary'] - quartiles['min_salary']) / (quartiles['max_salary'] - quartiles['max_salary']))  
  
quartiles.head()
```

```
Out[11]:
```

| | first_name | last_name | salary | min_salary | max_salary | Salary Range Position |
|---|------------|-----------|---------|------------|------------|-----------------------|
| 0 | Steven | King | 24000.0 | 20000.0 | 40000.0 | 0.20 |
| 1 | Neena | Kochhar | 17000.0 | 15000.0 | 30000.0 | 0.13 |
| 2 | Lex | De Haan | 17000.0 | 15000.0 | 30000.0 | 0.13 |
| 3 | Alexander | Hunold | 9000.0 | 4000.0 | 10000.0 | 0.83 |
| 4 | Bruce | Ernst | 6000.0 | 4000.0 | 10000.0 | 0.33 |

```
In [12]: #This function will be able to assign a quartile based on the position value (from 0 to 1) that we added  
def get_quartile(p):  
    if p <= 0.25:  
        return 'Q1'  
    if p <= 0.5:  
        return 'Q2'  
    if p <= 0.75:  
        return 'Q3'  
    else:  
        return 'Q4'  
  
#This applies the function to create new column, Salary Quartile  
quartiles['Salary Quartile'] = quartiles['Salary Range Position'].apply(get_quartile)  
  
quartiles.head()
```

Out[12]:

| | first_name | last_name | salary | min_salary | max_salary | Salary Range Position | Salary Quartile |
|---|------------|-----------|---------|------------|------------|-----------------------|-----------------|
| 0 | Steven | King | 24000.0 | 20000.0 | 40000.0 | 0.20 | Q1 |
| 1 | Neena | Kochhar | 17000.0 | 15000.0 | 30000.0 | 0.13 | Q1 |
| 2 | Lex | De Haan | 17000.0 | 15000.0 | 30000.0 | 0.13 | Q1 |
| 3 | Alexander | Hunold | 9000.0 | 4000.0 | 10000.0 | 0.83 | Q4 |
| 4 | Bruce | Ernst | 6000.0 | 4000.0 | 10000.0 | 0.33 | Q2 |

In [13]:

```
#Let's create a bar chart to see the distribution of employee salary among salary ranges
#First lets get a DataFrame of the counts of quartiles

quartile_counts = quartiles['Salary Quartile'].value_counts().reset_index()
quartile_counts.columns = ['Quartile', 'Frequency']

quartile_counts
```

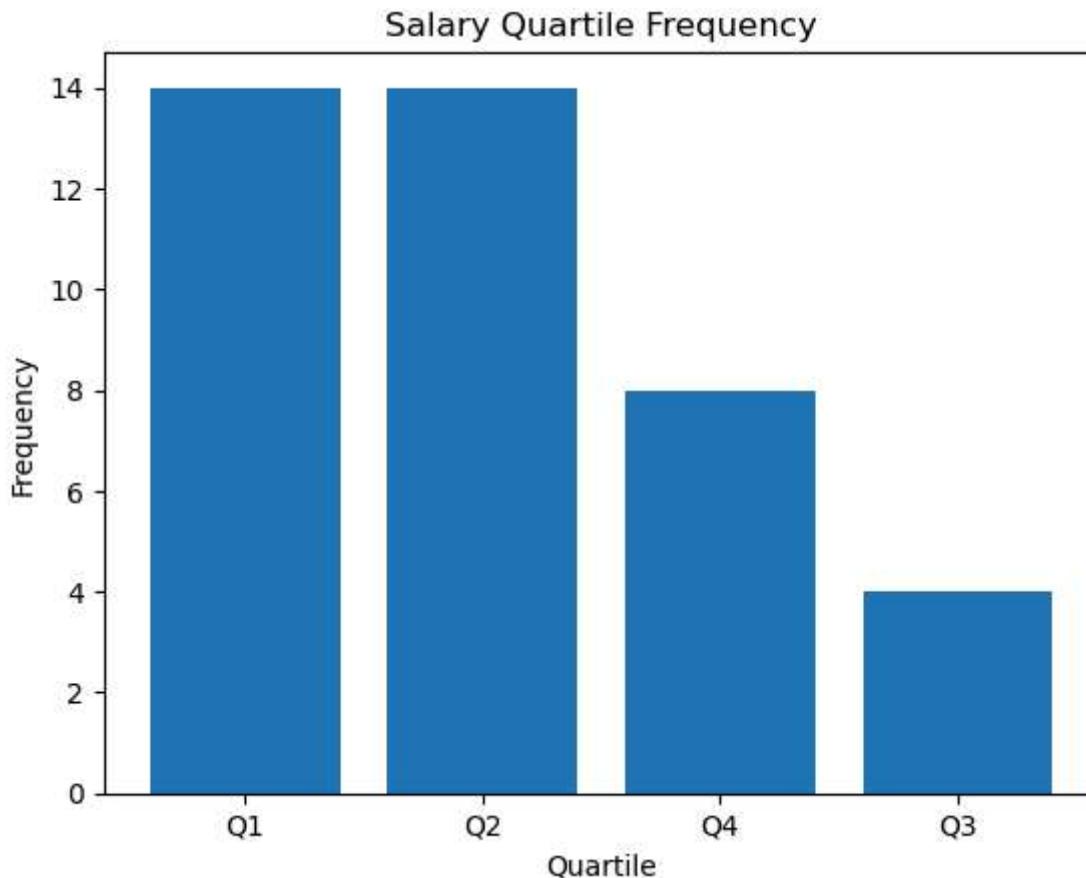
Out[13]:

| | Quartile | Frequency |
|---|----------|-----------|
| 0 | Q1 | 14 |
| 1 | Q2 | 14 |
| 2 | Q4 | 8 |
| 3 | Q3 | 4 |

In [14]:

```
#Now we have a DataFrame that can be easily visualized with matplotlib.plot
plt.bar(quartile_counts['Quartile'], quartile_counts['Frequency'])
plt.xlabel('Quartile')
plt.ylabel('Frequency')
plt.title('Salary Quartile Frequency')
plt.show()

#In the graph below, we can see that the majority of salaries fall within the lower two quartiles.
#Without including more variables to better understand this distribution, we cannot infer much.
#It could be different scenarios: high turnover so there are new employees near the salary range minimum, or
#or a lack of raises or career progression for employees
```



```
In [15]: #Let's now consider the tenure of an employee (in rounded years), in relation to their position  
#in their salary range. This will let us see if more tenured employees are further in the salary range.  
  
#Let's create a Dataframe that is the last name of the employee and the hire date  
  
q_years = """  
    SELECT last_name,  
          hire_date,  
          CAST ((julianday('now') - julianday(hire_date)) / 365 AS INT) AS years_employed  
    FROM employees;  
    """
```

```
df_years = pd.read_sql(q_years, conn)
df_years.head()
```

Out[15]:

| | last_name | hire_date | years_employed |
|---|-----------|------------|----------------|
| 0 | King | 1987-06-17 | 38 |
| 1 | Kochhar | 1989-09-21 | 35 |
| 2 | De Haan | 1993-01-13 | 32 |
| 3 | Hunold | 1990-01-03 | 35 |
| 4 | Ernst | 1991-05-21 | 34 |

In [17]:

```
#Let's combine the years_df to our quartiles DataFrame in order to then create a scatterplot
#with axes Years Employed and Salary Range Position

#Warning! This merge only works because there are no duplicate last names, otherwise
#this should be done using employee_id.

quartiles = quartiles.merge(df_years, on ='last_name', how='left')

quartiles.head()
```

Out[17]:

| | first_name | last_name | salary | min_salary | max_salary | Salary Range Position | Salary Quartile | hire_date | years_employed |
|---|------------|-----------|---------|------------|------------|-----------------------|-----------------|------------|----------------|
| 0 | Steven | King | 24000.0 | 20000.0 | 40000.0 | 0.20 | Q1 | 1987-06-17 | 38 |
| 1 | Neena | Kochhar | 17000.0 | 15000.0 | 30000.0 | 0.13 | Q1 | 1989-09-21 | 35 |
| 2 | Lex | De Haan | 17000.0 | 15000.0 | 30000.0 | 0.13 | Q1 | 1993-01-13 | 32 |
| 3 | Alexander | Hunold | 9000.0 | 4000.0 | 10000.0 | 0.83 | Q4 | 1990-01-03 | 35 |
| 4 | Bruce | Ernst | 6000.0 | 4000.0 | 10000.0 | 0.33 | Q2 | 1991-05-21 | 34 |

```
In [18]: #Before we create our scatterplot, let's look at the min, max, and mean age to give us more context into this organization

mean_years = quartiles['years_employed'].mean()
min_years = quartiles['years_employed'].min()
max_years = quartiles['years_employed'].max()

print("Mean Years Employed:", mean_years)
print("Minimum Years Employed:", min_years)
print("Maximum Years Employed:", max_years)

#As we can see, the employees of this organization are loyal! The Department of Labor
#reported that as of January 2024, the average stay for an employe is 3.9 years.
```

Mean Years Employed: 29.025

Minimum Years Employed: 25

Maximum Years Employed: 38

```
In [19]: #Now Let's create our scatterplot!
```

```
plt.scatter(quartiles['Salary Range Position'], quartiles['years_employed'])
plt.xlabel('Salary Range Position')
plt.ylabel('Years Employed')
plt.title('Salary Range Position Vs. Years Employed')

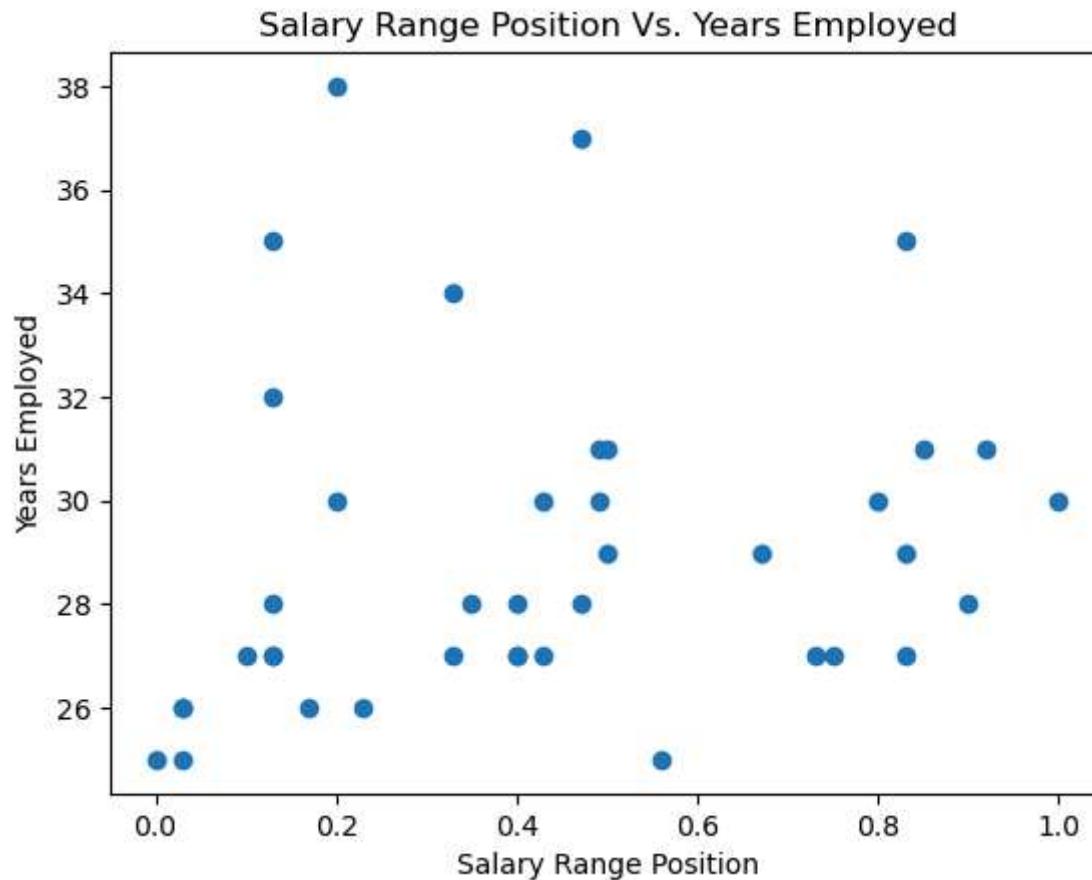
plt.show()
```

#Uh oh! This scatterplot does not evidence best practices for compensation strategies.

#As employees progress in a company, they should also progress in their salary range.

#Assuming a 30-year tenure, an employee who has a tenure of 15 years should

#be near the midpoint of their salary range. We are looking for a positive linear relationship.



```
In [20]: #Let's return a filtered dataframe where employees have been with the organization more than 15 years
# are not past their salary ranges midpoint

not_at_midpoint_df = quartiles[(quartiles['years_employed'] >= 15) & (quartiles['Salary Range Position'] < 0.5)]
num_underpaid = len(not_at_midpoint_df['years_employed'])

#Please be cautious of below line. Coder knows that number of employees is 40, however this would change
#when employees leave or new employees are hired.
print((num_underpaid / 40)*100)

#The output shows us that 65% of employees are likely being underpaid based on their specific salary range and
#the years they have worked with the organization.
```

65.0

```
In [21]: #Let's use a Linear regression to see how strongly years of experience can predict salary.  
#We will need to import from the statsmodels.api module  
import statsmodels.api as sm  
  
#Initializing predictor (X) and target (y)  
X = quartiles['years_employed']  
y = quartiles['salary']  
  
#Adds constant  
X = sm.add_constant(X)  
  
#Fits the model  
model = sm.OLS(y, X).fit()  
  
print(model.summary())  
  
#To interpret the Intercept, whenever an employee has zero (0) years of experience, then this linear model predicts  
#that the employee will have a salary of -$14,526. This result is misleading, as the average years employed is 28,  
#meaning that 0 is far from this average, which is why the model outputs a negative intercept.  
  
#The coefficient of years employed (778.8) indicates how much is earned with each additional year of experience,  
#which the model predicts to be $778 for each added year employed.  
  
#We also see that the Adjusted R-squared, or how much the dependent variable changes based on the predictor,  
#is 0.279. This indicates that 27.9% of change in salary can be explained by years employed. For the p-value,  
#we have a value < 0.000, which indicates that years employed is very likely to have a non-zero impact  
#on salary. Overall, this model is lacking in explanatory power, due to the low Adjusted R-squared value.  
#The model could be improved by adding other variables, such as location, department, or even manager.
```

OLS Regression Results

```
=====
Dep. Variable: salary R-squared:      0.298
Model: OLS   Adj. R-squared:      0.280
Method: Least Squares F-statistic:    16.13
Date: Mon, 14 Jul 2025 Prob (F-statistic): 0.000269
Time: 12:51:43 Log-Likelihood: -386.40
No. Observations: 40 AIC:            776.8
Df Residuals:     38 BIC:            780.2
Df Model:         1
Covariance Type: nonrobust
=====
          coef    std err      t      P>|t|      [0.025      0.975]
-----
const      -1.46e+04  5675.525   -2.572     0.014    -2.61e+04   -3110.231
years_employed  780.6971  194.386     4.016     0.000      387.183    1174.211
=====
Omnibus:           0.496 Durbin-Watson:       1.293
Prob(Omnibus):    0.781 Jarque-Bera (JB):  0.059
Skew:              0.048 Prob(JB):        0.971
Kurtosis:          3.160 Cond. No.       270.
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In []: