1.Importing necssary libraries

(安装Python与SQL连接的桥梁)

In [1]:
```
#安装sqlalchemy模块
#Installera sqlalchemy-modulen
!pip install sqlalchemy
!pip install pyodbc
```

```
Requirement already satisfied: sqlalchemy in c:\users\armen\anaconda3\lib\site-packa
ges (1.4.39)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\armen\anaconda3\lib\site
-packages (from sqlalchemy) (2.0.1)
Requirement already satisfied: pyodbc in c:\users\armen\anaconda3\lib\site-packages
(4.0.34)
```

2.Creating engine

数据库连接引擎组件
导入库并创建引擎

In [147…
```
# 在sqlalchemy安装模块后，‘create_engine’用于建立与数据库的连接，
# Efter installation av sqlalchemy-modulen används 'create_engine' för att upprätta 
# ‘MetaData’元数据，用于反映数据库的结构信息，‘Table’表，代表数据库中的一个表，
# ‘Inspect’检查，用于获取数据库的详细信息

from sqlalchemy import create_engine, MetaData,Table,inspect

#引入一个数据分析和处理的库，令其别名是pd.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [144…
```
#定义一个函数Define a function，接受数据库方言、服务器地址、数据库名，可选的用户和密
#Definiera en funktion som accepterar databasdialekt, serveradress, databasnamn,
#valfri användare och lösenord och om integrerade säkerhetsverifieringsflaggor ska a

def new_engine(dialect, server, database, user=None, password=None, integrated_secu
    #如果启用了集成安全验证（即Windows验证），则使用以下格式的连接字符串。
    if integrated_security:
        # For Windws authentication # 使用Windows验证创建连接字符串
        # For SQL Server authentication #如果不使用集成安全验证，则使用SQL Server验证
        eng = f"{dialect}://{server}/{database}?trusted_connection=yes&driver=ODBC+D
    else:
        # 使用SQL Server验证创建连接字符串。
        eng = f"{dialect}://{user}:{password}@{server}/{database}?driver=ODBC+Driver
    #打印出创建的连接字符串
    print(eng)

    return create_engine(eng)
```

In [148…
```
# 建立一个到指定SQL Server数据库的连接
# Upprätta en anslutning till den angivna SQL Server-databasen
engine = new_engine('mssql','DESKTOP-99SOD5T','AdventureWorks2022', integrated_secur
mssql://DESKTOP-99SOD5T/AdventureWorks2022?trusted_connection=yes&driver=ODBC+Driver
+17+for+SQL+Server
```

In [149…
```
#用于调试，以确认engine已正确创建 # Används för felsökning för att bekräfta att motor
print(type(engine))
```

```
<class 'sqlalchemy.engine.base.Engine'>
```

3. Query Databaseen

查看数据库所有表

In [150…
```
# 使用engine对象来建立与数据库的连接，并将这个连接赋值给变量connection。
# Använd motorobjektet för att upprätta en anslutning till databasen och tilldela den
connection = engine.connect()
```

In [151…
```
#验证，确保connection对象已经被正确创建并且是预期的类型
#Kontrollera att anslutningsobjektet har skapats korrekt och är av den förväntade typ
print(type(connection))
```

```
<class 'sqlalchemy.engine.base.Connection'>
```

In [152…
```
#创建一个Inspector对象，用于从提供的engine（数据库连接引擎）获取数据库的结构和细节信
# Skapa ett Inspector-objekt för att hämta strukturen och detaljerad information om d
inspector = inspect(engine)


# 使用Inspector对象获取数据库中所有模式（schemas）的名称。
# Använd Inspector-objektet för att få namnen på alla scheman i databasen.
schemas = inspector.get_schema_names()

# 打印出数据库中的模式列表 Skriv ut en lista över scheman i databasen
print(schemas)
```

```
['db_accessadmin', 'db_backupoperator', 'db_datareader', 'db_datawriter', 'db_ddladm
in', 'db_denydatareader', 'db_denydatawriter', 'db_owner', 'db_securityadmin', 'db
o', 'guest', 'HumanResources', 'INFORMATION_SCHEMA', 'Person', 'Production', 'Purcha
sing', 'Sales', 'sys']
```

In [165…
```
# 遍历schemas列表，即逐个处理列表中的每个模式
# Läsa schemalistan, d.v.s. bearbeta varje schema i listan ett efter ett

for schema in schemas:
    print(schema)
```

```
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
dbo
guest
HumanResources
INFORMATION_SCHEMA
Person
Production
Purchasing
Sales
sys
```

4.Välj önskad lista och läs informationen i listan

选取所需列表，并读取列表中的数据

In [169…

```
pd.set_option('display.max_rows', 10)
```

In [170…

```
# 查询Person表中的业务实体ID、名字和姓氏，并按业务实体ID排序
# Fråga företagsenhets-ID, förnamn och efternamn i tabellen Person och sortera efter
query_person = """
SELECT BusinessEntityID, FirstName, LastName
FROM Person.Person
ORDER BY BusinessEntityID
"""
person_data = pd.read_sql(query_person, engine)
person_data
```

Out[170]:

	BusinessEntityID	FirstName	LastName
0	1	Ken	Sánchez
1	2	Terri	Duffy
2	3	Roberto	Tamburello
3	4	Rob	Walters
4	5	Gail	Erickson
...	...	...	...
19967	20773	Crystal	Guo
19968	20774	Isabella	Richardson
19969	20775	Crystal	He
19970	20776	Crystal	Zheng
19971	20777	Crystal	Hu

19972 rows × 3 columns

In [171…

```
# 查询HumanResources.Employee表中前100个业务实体ID和职位标题，并按职位标题排序
# Fråga de 100 bästa företagsenhets-id:n och jobbtitlarna i tabellen HumanResources.I

query_employee = """
SELECT TOP 100 BusinessEntityID, JobTitle
FROM HumanResources.Employee
ORDER BY JobTitle
"""
employee_data = pd.read_sql(query_employee, engine)
employee_data
```

Out[171]:

	BusinessEntityID	JobTitle
0	245	Accountant
1	248	Accountant
2	241	Accounts Manager
3	246	Accounts Payable Specialist
4	247	Accounts Payable Specialist
...	...	...
95	90	Production Technician - WC10
96	89	Production Technician - WC10
97	88	Production Technician - WC10
98	103	Production Technician - WC10
99	104	Production Technician - WC10

100 rows × 2 columns

In [172…

```
# 结合HumanResources.Employee和Person.Person两个表的数据，按业务实体ID排序
# Kombinera data från tabellerna HumanResources.Employee och Person.Person och sorte
# INNER JOIN在这里用于关联两个表中相同的BusinessEntityID
# INNER JOIN används här för att associera samma BusinessEntityID i två tabeller

query_combined = """
SELECT E.BusinessEntityID, P.FirstName, P.LastName, E.JobTitle
FROM HumanResources.Employee AS E
INNER JOIN Person.Person AS P ON E.BusinessEntityID = P.BusinessEntityID
ORDER BY BusinessEntityID
"""
combined_data = pd.read_sql(query_combined, engine)
combined_data
```

Out[172]:

	BusinessEntityID	FirstName	LastName	JobTitle
0	1	Ken	Sánchez	Chief Executive Officer
1	2	Terri	Duffy	Vice President of Engineering
2	3	Roberto	Tamburello	Engineering Manager
3	4	Rob	Walters	Senior Tool Designer
4	5	Gail	Erickson	Design Engineer
...	...	...	...	...
285	286	Lynn	Tsoflias	Sales Representative
286	287	Amy	Alberts	European Sales Manager
287	288	Rachel	Valdez	Sales Representative
288	289	Jae	Pak	Sales Representative
289	290	Ranjit	Varkey Chudukatil	Sales Representative

290 rows × 4 columns

In [173…

```
# 定义SQL查询，从HumanResources.vEmployee视图选择四个字段：
# Definiera en SQL-fråga och välj fyra fält från HumanResources.vEmployee-vyn:
# BusinessEntityID（员工编号），FirstName（名），LastName（姓），JobTitle（职称）
# BusinessEntityID (employee number), FirstName (first name), LastName (last name), 

query_vEmployee = """
SELECT BusinessEntityID, FirstName, LastName, JobTitle
FROM AdventureWorks2022.HumanResources.vEmployee
ORDER BY JobTitle  -- 按员工职务升序排序
"""
# 使用Pandas的read_sql函数执行SQL查询，并将结果存储在vEmployee_data DataFrame中
## Använd Pandas read_sql-funktion för att köra SQL-frågor och lagra resultaten i vE
vEmployee_data = pd.read_sql(query_vEmployee, engine)

# 显示查询结果 # Visa frågeresultat
vEmployee_data
```

Out[173]:

	BusinessEntityID	FirstName	LastName	JobTitle
0	245	Barbara	Moreland	Accountant
1	248	Mike	Seamans	Accountant
2	241	David	Liu	Accounts Manager
3	246	Dragan	Tomic	Accounts Payable Specialist
4	247	Janet	Sheperdigian	Accounts Payable Specialist
...	...	...	...	...
285	13	Janice	Galvin	Tool Designer
286	12	Thierry	D'Hers	Tool Designer
287	2	Terri	Duffy	Vice President of Engineering
288	25	James	Hamilton	Vice President of Production
289	273	Brian	Welcker	Vice President of Sales

290 rows × 4 columns

In [158…

```
# 执行 SQL 查询
# Kör SQL-fråga

query = """
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_CATALOG = 'AdventureWorks2022' AND (TABLE_NAME LIKE '%Pay%' OR TABLE_NAMI
"""

# 使用 pandas 的 read_sql 函数运行 SQL 查询并获取结果
# Använd funktionen read_sql för pandas för att köra SQL-frågan och få resultaten
result = pd.read_sql(query, engine)
result
```

Out[158]:

	TABLE_NAME
0	EmployeePayHistory

In [159…

```
# 尝试从数据库中查询员工的薪资历史记录
# Försök att fråga den anställdes lönehistorik från databasen

query_pay_history = """
SELECT TOP 10 BusinessEntityID, RateChangeDate, Rate, PayFrequency
FROM HumanResources.EmployeePayHistory
ORDER BY BusinessEntityID
"""
pay_history_data = pd.read_sql(query_pay_history, engine)
pay_history_data
```

Out[159]:

	BusinessEntityID	RateChangeDate	Rate	PayFrequency
0	1	2009-01-14	125.5000	2
1	2	2008-01-31	63.4615	2
2	3	2007-11-11	43.2692	2
3	4	2007-12-05	8.6200	2
4	4	2010-05-31	23.7200	2
5	4	2011-12-15	29.8462	2
6	5	2008-01-06	32.6923	2
7	6	2008-01-24	32.6923	2
8	7	2009-02-08	50.4808	2
9	8	2008-12-29	40.8654	2

In [160…

```
# 合并vEmployee_data和pay_history_data，基于共同的'BusinessEntityID'
# Slå samman vEmployee_data och pay_history_data baserat på vanligt "BusinessEntityII
combined_data = pd.merge(vEmployee_data, pay_history_data, on='BusinessEntityID', ho

# 显示合并后的数据
#Visa sammanslagna data
combined_data
```

Out[160]:

	BusinessEntityID	FirstName	LastName	JobTitle	RateChangeDate	Rate	PayFrequency
0	1	Ken	Sánchez	Chief Executive Officer	2009-01-14	125.5000	2
1	6	Jossef	Goldberg	Design Engineer	2008-01-24	32.6923	2
2	5	Gail	Erickson	Design Engineer	2008-01-06	32.6923	2
3	3	Roberto	Tamburello	Engineering Manager	2007-11-11	43.2692	2
4	8	Diane	Margheim	Research and Development Engineer	2008-12-29	40.8654	2
5	7	Dylan	Miller	Research and Development Manager	2009-02-08	50.4808	2
6	4	Rob	Walters	Senior Tool Designer	2007-12-05	8.6200	2
7	4	Rob	Walters	Senior Tool Designer	2010-05-31	23.7200	2
8	4	Rob	Walters	Senior Tool Designer	2011-12-15	29.8462	2
9	2	Terri	Duffy	Vice President of Engineering	2008-01-31	63.4615	2

In [174…

```
# 定义SQL查询
#Definiera SQL-fråga
query = """
SELECT
    E.BusinessEntityID,       -- 选择EmployeePayHistory表的员工编号
    V.FirstName,              -- 选择vEmployeeDepartment视图的名字
    V.LastName,               -- 选择vEmployeeDepartment视图的姓氏
    V.JobTitle,               -- 选择vEmployeeDepartment视图的工作职称
    V.Department,             -- 选择vEmployeeDepartment视图的部门
    E.RateChangeDate,         -- 选择EmployeePayHistory表的薪资变动日期
    E.Rate,                   -- 选择EmployeePayHistory表的薪资率
    E.PayFrequency            -- 选择EmployeePayHistory表的薪资支付频率
FROM
    HumanResources.EmployeePayHistory AS E   -- 从EmployeePayHistory表中获取数据
INNER JOIN
    AdventureWorks2022.HumanResources.vEmployeeDepartment AS V  -- 与vEmployeeDepartı
ON
    E.BusinessEntityID = V.BusinessEntityID  -- 根据BusinessEntityID进行联合
ORDER BY
    E.BusinessEntityID;                      -- 按BusinessEntityID排序

"""


# 使用Pandas的read_sql函数执行SQL查询，并将结果存储在DataFrame中
## Använd Pandas read_sql-funktion för att köra SQL-frågor och lagra resultaten i en
combined_data = pd.read_sql(query, engine)

# 显示查询结果
combined_data
```

Out[174]:

	BusinessEntityID	FirstName	LastName	JobTitle	Department	RateChangeDate	Ra
0	1	Ken	Sánchez	Chief Executive Officer	Executive	2009-01-14	125.50
1	2	Terri	Duffy	Vice President of Engineering	Engineering	2008-01-31	63.46
2	3	Roberto	Tamburello	Engineering Manager	Engineering	2007-11-11	43.26
3	4	Rob	Walters	Senior Tool Designer	Tool Design	2007-12-05	8.62
4	4	Rob	Walters	Senior Tool Designer	Tool Design	2010-05-31	23.72
...	...	...	...	...	...	...	
311	286	Lynn	Tsoflias	Sales Representative	Sales	2013-05-30	23.07
312	287	Amy	Alberts	European Sales Manager	Sales	2012-04-16	48.10
313	288	Rachel	Valdez	Sales Representative	Sales	2013-05-30	23.07
314	289	Jae	Pak	Sales Representative	Sales	2012-05-30	23.07
315	290	Ranjit	Varkey Chudukatil	Sales Representative	Sales	2012-05-30	23.07

316 rows × 8 columns

5.Extrahera nödvändiga data och gör lönekonfidensintervallanalys

提取所需要数据，做工资置信区间分析

In [162…

```
# För testa 提取，并查询， Research and Development 部门的工资水平
# # För testa Ta ut och fråga lönenivån på forsknings- och utvecklingsavdelningen

query = """
SELECT
    E.BusinessEntityID,
    V.FirstName,
    V.LastName,
    V.JobTitle,
    V.Department,
    E.RateChangeDate,
    E.Rate,
    E.PayFrequency
FROM
    HumanResources.EmployeePayHistory AS E
INNER JOIN
    AdventureWorks2022.HumanResources.vEmployeeDepartment AS V
ON
    E.BusinessEntityID = V.BusinessEntityID
```

```
WHERE
    V.Department = 'Research and Development' -- 过滤条件，只选择研发部门的员工
ORDER BY
    E.BusinessEntityID;

"""

# 使用Pandas的read_sql函数执行SQL查询，并将结果存储在DataFrame中
# Använd Pandas read_sql-funktion för att köra SQL-frågor och lagra resultaten i en I
combined_data = pd.read_sql(query, engine)

# 显示查询结果
combined_data
```

Out[162]:

	BusinessEntityID	FirstName	LastName	JobTitle	Department	RateChangeDate	Rate	I
0	7	Dylan	Miller	Research and Development Manager	Research and Development	2009-02-08	50.4808	
1	8	Diane	Margheim	Research and Development Engineer	Research and Development	2008-12-29	40.8654	
2	9	Gigi	Matthew	Research and Development Engineer	Research and Development	2009-01-16	40.8654	
3	10	Michael	Raheem	Research and Development Manager	Research and Development	2009-05-03	42.4808	

In [166…

```
# För testa 在提取，并查询， Research and Development 部门的工资水平后，计算这个部门
#För testa extraherar och ifrågasätter lönenivån på forsknings- och utvecklingsavdel
#och beräknar sedan konfidensintervallet för inkomsterna för anställda på denna avde

# 从查询结果中提取工资数据
# Extrahera lönedata från frågeresultat
salaries = r_and_d_data['Rate'].tolist()

# 计算均值和标准误差
# Beräkna medelvärde och standardfel
mean = np.mean(salaries)
std_err = stats.sem(salaries)

# 定义置信水平
#Definiera konfidensnivå
confidence = 0.95

# 计算置信区间
# Beräkna konfidensintervall
interval = stats.t.interval(confidence, len(salaries)-1, loc=mean, scale=std_err)

print(f"{confidence*100}% confidence is {interval}")
```

```
95.0% confidence är (36.350434825484925, 50.995765174515086)
```

In [168…

```
# 提取所有员工的工资数据
# Extrahera lönedata för alla anställda
all_salaries = combined_data['Rate'].tolist()

# 检查是否有足够的数据点
# Kontrollera om det finns tillräckligt med datapunkter
```

```
if len(all_salaries) > 1:
    # 计算均值和标准误差 # Beräkna medelvärde och standardfel
    mean = np.mean(all_salaries)
    std_err = stats.sem(all_salaries)

    # 定义置信水平 # Definiera konfidensnivå
    confidence = 0.95

    # 计算置信区间 # Beräkna konfidensintervall
    interval = stats.t.interval(confidence, len(all_salaries)-1, loc=mean, scale=std

    # 打印结果 # Skriv ut resultat
    print(f"All department : {confidence*100}% confidence is {interval}")
else:
    print("Not enough data points to calculate confidence interval")
```

```
All department : 95.0% confidence is (36.350434825484925, 50.995765174515086)
```

```
In [ ]:
```