Teoretiska Frågor

1. Beskriv kort hur en relationsdatabas fungerar.
Svar:
Relationsdatabaser fungerar genom att lagra data i tabeller, som var och en består av rader och kolumner.
Varje rad representerar en datapost och varje kolumn representerar ett specifikt attribut för data.
Dessa tabeller är relaterade till varandra genom nyckelord som primärnycklar och främmande nycklar. En primärnyckel är ett fält i en tabell som unikt identifierar varje post, medan en främmande nyckel är ett fält i en tabell som refererar till primärnyckeln för en annan tabell.
Relationsdatabaser använder SQL (Structured Query Language) för att utföra frågor, infoga, uppdatera och radera data.

2. Vad menas med ”CRUD” flödet?
Svar:
CRUD-flöden i databaser och applikationer inkluderar fyra underliggande datamanipuleringsoperationer, som omfattar:
Creat: Lägg till ny data till databasen.
Read: Hämta och hämta information från databasen.
Update: Ändra befintliga data i databasen.
Delete: Ta bort data från databasen.
Då, denna process återspeglar sedan flödet av dataoperationer i programmet, från skapande till radering eller uppdatering, och är kärnan i datadrivna applikationer.

3. Beskriv kort vad en ”left join” och ”inner join” är. Varför använder man det?
Svar:
"Left Join" den returnerar alla rader från vänstra (första) tabellen och matchade rader från högra (andra) tabellen. Om det inte finns någon matchning, returneras null-värden för högra tabellen.

"Inner Join" i en databas innebär att endast de rader där det finns matchningar i båda tabellernas angivna kolumner inkluderas i det slutliga sökresultatet. Om en rad matchar i en tabell men inte i den andra, kommer den raden inte att visas i resultatet. Enkelt uttryckt, en "Inner Join" returnerar den del som matchar mellan två tabeller.

4. Beskriv kort vad indexering i SQL innebär.
Svar:
I SQL är ett index en datastruktur som används för att förbättra effektiviteten i databasfrågor. Det gör att databassystemet snabbt kan hitta specifika data i tabellen, liknande innehållsförteckningen i en bok. Index är särskilt användbara för stora databaser där det skulle vara mycket tidskrävande att söka på alla rader direkt. Även om index kan påskynda frågor avsevärt, tar de också upp ytterligare lagringsutrymme och kan vara mindre effektiva när data läggs till, raderas eller uppdateras eftersom själva indexen behöver uppdateras. Kort sagt är index ett viktigt verktyg för databasoptimering.

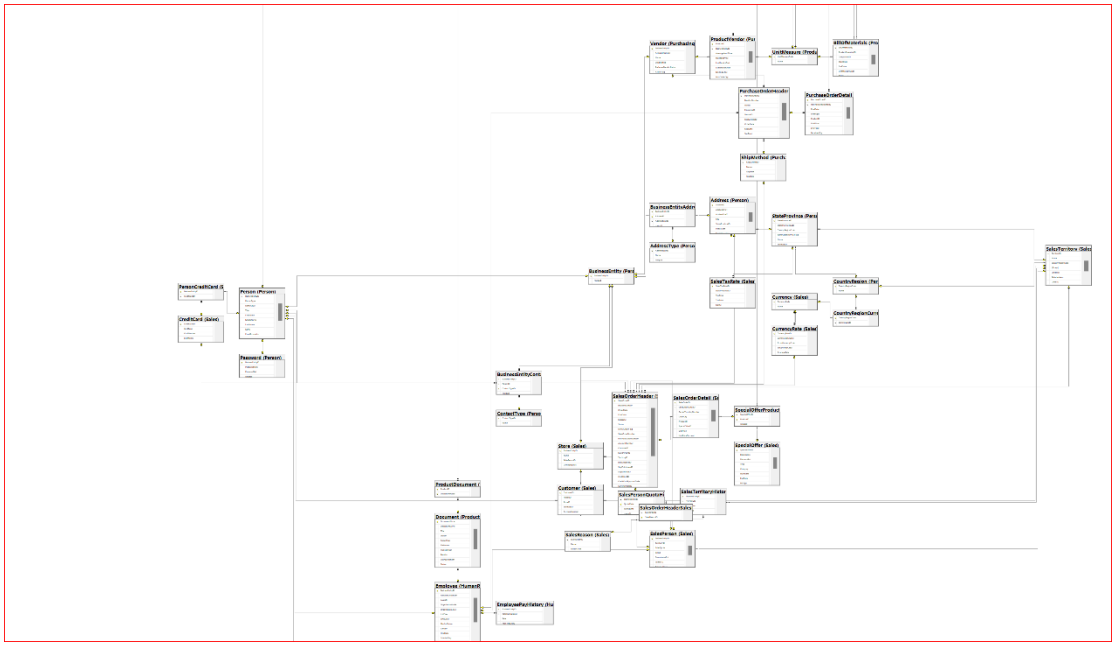
5. Beskriv kort vad en vy i SQL är.
Svar:
En vy i SQL (kallad "view" på engelska) är en virtuell tabell som skapas genom en SQL-fråga på en eller flera tabeller. Vyn innehåller ingen data i sig utan är en sparad fråga som när den körs, hämtar data från de underliggande tabellerna. Vyerna används för att förenkla komplexa frågor, förbättra säkerheten genom att begränsa åtkomst till specifika data, och för att ge en mer anpassad vy av databasen till olika användare. Vyerna uppdateras dynamiskt när de underliggande tabellernas data ändras.

6. Beskriv kort vad en lagrad procedur i SQL är
Svar:
Lagrade procedurer i SQL motsvarar en uppsättning SQL-instruktioner som kan köras flera gånger, liknande funktioner i programmering. Man kan kombinera en serie komplexa SQL-kommandon till en lagrad procedur och spara proceduren i databasen. När du behöver utföra dessa kommandon, ring bara denna lagrade procedur. Fördelen med detta är att det förbättrar återanvändbarheten av koden och minskar arbetsbelastningen med att upprepade gånger skriva samma kod. Det hjälper också till att förbättra exekveringseffektiviteten och datasäkerheten.

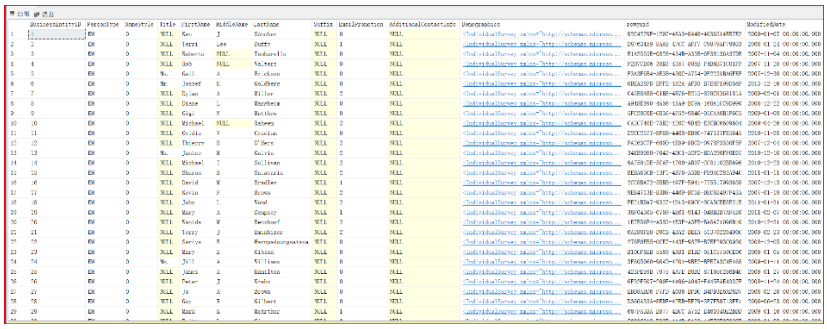
Rapport

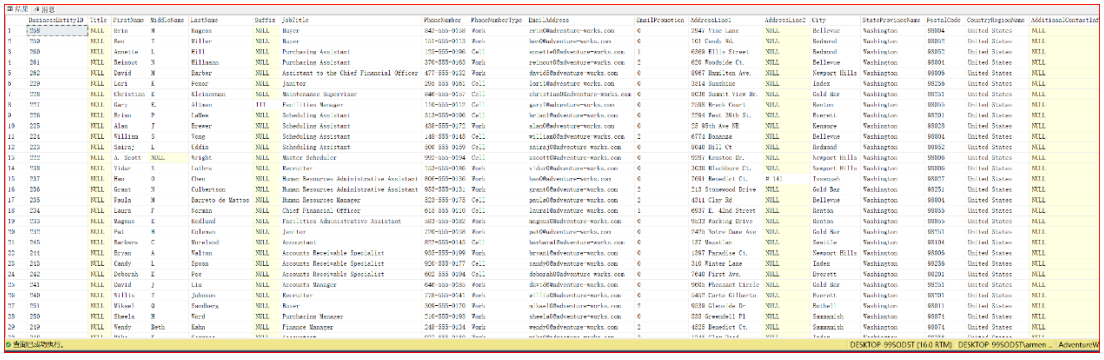
1.AdventureWorks2022-databasen är en komplex företagsdatabas som innehåller nyckeltabeller inom försäljning, personalresurser och produktion. Dess struktur är högt normaliserad med interrelationer mellan tabeller som stöder komplexa frågor och analyser.

2.När jag först fick tillgång till databasen, fokuserade jag på att förstå dess innehåll genom att granska databasens relationsschema i SSMS.
Example:



Sedan kommer jag att öppna tabellen jag behöver i SSMS och se innehållet i kolumnerna och raderna i varje lista.
Example:





Sedan kommer jag att jämföra innehållet i varje lista, tänka på förhållandet mellan listorna och fundera på vilken data jag behöver fråga och extrahera.

Jag valde att analysera "anställdas lönekonfidensintervaller" som ett intressant område. Jag beräknade lönekonfidensintervaller för alla avdelningar, särskilt för Research and Development, med hjälp av SQL-frågor och Python-beräkningar. Jag definierade ett 95 % konfidensintervall, vilket innebär att det är 95 % sannolikt att det verkliga genomsnittet av löner ligger inom detta intervall.

Resultaten:
Research and Development, 95,0 % konfidensintervall är (36,350434825484925, 50,995765174515086)

Alla sektorer 95,0 % konfidensintervall är (36,350434825484925, 50,995765174515086)

3.Resultaten visar att konfidensintervallen för både Research and Development avdelningen och alla avdelningar är nästan lika, vilket antyder att det inte finns någon betydande skillnad i löner mellan avdelningarna eller att urvalsstorleken inte är tillräckligt stor för att visa en skillnad.

4. Executive Summary:
Analysprocessen för databasen AdventureWorks2022:

1. Först förstår jag databasstrukturen genom databasrelationsdiagrammet, och väljer sedan vad ska jag beräkna för konfidensintervall.
2. Jag använde SQL-Query för att extrahera lönedata för anställda på Research and Development -avdelningen och alla avdelningar på hela företaget och genomförde sedan statistisk analys i Python-miljön.
3. Min analysidé är att först hitta målet som ska analyseras, vilket betyder målet för konfidensintervallet.
4. Fråga sedan vilka listor eller tabeller som innehåller den data jag behöver, vilket innebär att query tabellen.
5. Sedan frågar jag i formuläret för att extrahera den data jag behöver.
6. Slutligen, efter att ha erhållit data, utför konfidensintervallanalys. Data som är medarbetarens nummer, namn, efternamn, titel och avdelning.
7. Genom dessa steg har vi beräknat ett 95 % konfidensintervall för anställdas löner.
   Den här analysen visar inte bara upp vår datakrossande förmåga, utan ger också en djupgående förståelse för företagets ersättningsstruktur. Detta är avgörande för att optimera personalhanteringen och utveckla ersättningsstrategier.

1. **Utmaningar du haft under arbetet samt hur du hanterat dem.**

   Det är första gången jag kommer i kontakt med en databas och analyserar den. Det är en utmaning i sig. När jag först öppnade de många listorna i databasen kände jag mig väldigt förvirrad och hade ingen aning om var jag skulle börja.
   Men genom att titta på videon från kursen och följa hur SSMS och Python fungerar i videon, som att skala en lök, löses problemen steg för steg. Nu förstår jag principerna, stegen och till och med analysidéerna för dataanalys. Den största utmaningen är att jag inte är skicklig i syntaxen för SSMS och Python-koder. Utan vägledningen från undervisningsvideor är det mycket svårt att slutföra en dataanalys.

2. **Vilket betyg du anser att du skall ha och varför**

   Jag tror att jag bara kan få G, eftersom jag inte är bekant med kodsyntax. Jag behöver söka information på Internet för att slutföra kodskrivning, och jag kan inte skriva kod helt självständigt.

3. **Tips du hade ”gett till dig själv” i början av kursen nu när du slutfört den**

   Jag är fortfarande väldigt intresserad av dataanalys, jag känner att grunden för kodskrivning inte är tillräckligt stark, så om i början bör jag hänvisa mer till andras kodskrivning och lära mig mer om olika idéerna med dataanalys.

Kodningsprocess

1. Installera SQLalchemy-libraries.

Eftersom SQL-servern bara är en plattform för att ladda databaser måste vi använda Python under Jupyter för att analysera data, så vi installerar först sqlalchemy- och pyodbc-bibliotek i Python-språkmiljön för att göra det möjligt för Python-kod att driva databasen, och 'pyodbc ' är en specifik databasdrivrutin som gör att Python kan ansluta till en mängd olika databaser via ODBC-gränssnittet.

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-packages (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

Jag ska importera den skapande motorfunktionen och flera nyckelkomponenter från SQLAlchemy för att skapa en databasmotor. Detta är grunden för kommunikationen mellan SQLAlchemy och databasen.
Jag måste tillhandahålla databasanslutningsinformationen för att skapa denna motor.
"MetaData" används för att lagra tabeller och kolumner i databasstrukturen;
"Tabell" kan användas för att komma åt och manipulera data i tabellen.
inspektera: Denna funktion används för att få detaljerad information om databasen, såsom tabellnamn, kolumnnamn, etc.
Sedan måste också skapa en motor som ansluter till en specifik SQL Server-databas.Den skapade motorn ansluter till databasen med namnet AdventureWorks2022, som finns på servern DESKTOP-99SOD5T.

2.Creating engine 数据库连接引擎组件

导入库并创建引擎

```
In [2]: # 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 en anslutning till databasen,
        # ‘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，可选的用户和密码，以及是否使用集成安全验证的标志
          #Definiera en funktion som accepterar databasdialekt, serveradress, databasnamn,
          #valfri användare och lösenord och om integrerade säkerhetsverifieringsflaggor ska användas

          def new_engine(dialect, server, database, user=None, password=None, integrated_security=True):
              #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+Driver+17+for+SQL+Server"
              else:
                  # 使用SQL Server验证创建连接字符串。
                  eng = f"{dialect}://{user}:{password}@{server}/{database}?driver=ODBC+Driver+17+for+SQL+Server"
              #打印出创建的连接字符串
              print(eng)

              return create_engine(eng)
```

```
In [4]: # 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_security=True)
```

```
mssql://DESKTOP-99SOD5T/AdventureWorks2022?trusted_connection=yes&driver=ODBC+Driver+17+for+SQL+Server
```

```
In [5]: #用于调试，以确认engine已正确创建 # Används för felsökning för att bekräfta att motorn har skapats korrekt
        print(type(engine))
```

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

3. Query Databassen

Använd engine objektet för att upprätta en anslutning till databasen och tilldela denna anslutning till variabel Connection.

Skapa ett Inspector-objekt för att hämta strukturen och detaljerad information om databasen från den tillhandahållna motorn (databasanslutningsmotor).

Läs sedan schemalistan i databasen och läs varje schema i listan.

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 denna anslutning till variabelanslutningen.
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 typen
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息。
# Skapa ett Inspector-objekt för att hämta strukturen och detaljerad information om databasen från den tillhandahållna motorn (databasanslu
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_ddladmin', 'db_denydatareader', 'db_denydatawriter', 'db_ow
ner', 'db_securityadmin', 'dbo', 'guest', 'HumanResources', 'INFORMATION_SCHEMA', 'Person', 'Production', 'Purchasing', 'Sales', 'sys']
```

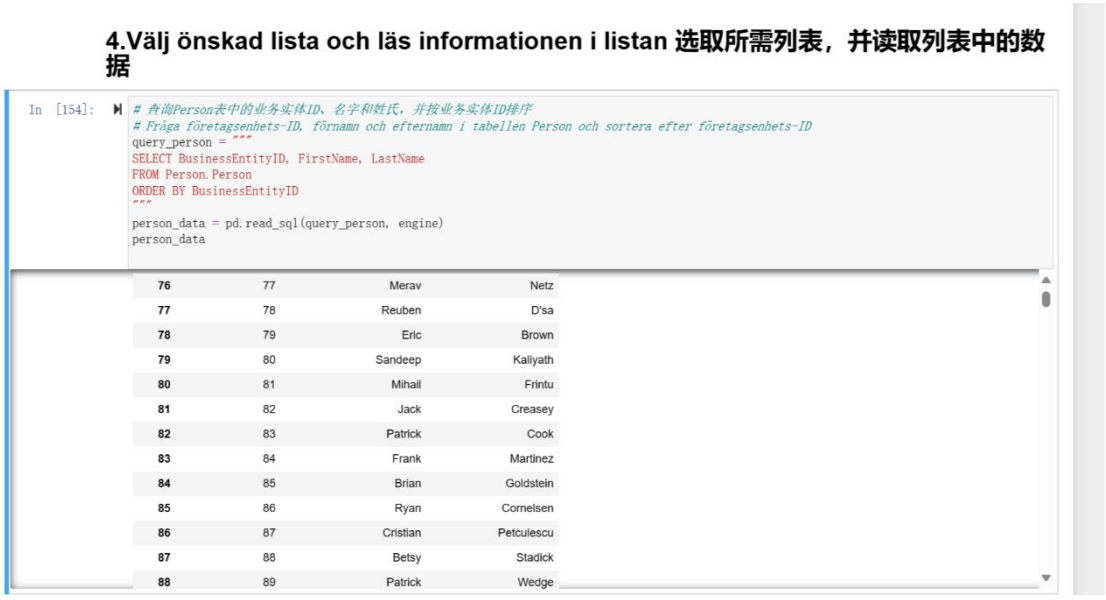
In [153]:
```
# 遍历schemas列表，即逐个处理列表中的每个模式
# Läsa schemalistan, d.v.s. bearbeta varje schema i listan ett efter ett

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 från frågeresultaten och läs data i listan

4-a. Fråga tabellen person.person och läs informationen BusinessEntityID, FirstName och LastName.



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 [154]:
```
# 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 företagsenhets-ID
query_person = """
SELECT BusinessEntityID, FirstName, LastName
FROM Person.Person
ORDER BY BusinessEntityID
"""

person_data = pd.read_sql(query_person, engine)
person_data
```

76	77	Merav	Netz
77	78	Reuben	D'sa
78	79	Eric	Brown
79	80	Sandeep	Kaliyath
80	81	Mihail	Frintu
81	82	Jack	Creasey
82	83	Patrick	Cook
83	84	Frank	Martinez
84	85	Brian	Goldstein
85	86	Ryan	Cornelsen
86	87	Cristian	Petculescu
87	88	Betsy	Stadick
88	89	Patrick	Wedge

4-b Fråga HumanResources.Employee och läs informationen om BusinessEntityID och JobTitle i listan

```
In [155]:
# 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.Employee och sortera efter jobbtitel

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

16	253	Buyer
17	254	Buyer
18	255	Buyer
19	256	Buyer
20	257	Buyer
21	258	Buyer
22	259	Buyer
23	1	Chief Executive Officer
24	234	Chief Financial Officer
25	218	Control Specialist
26	221	Control Specialist
27	270	Database Administrator
28	271	Database Administrator

4-c Kombinera data från tabellerna HumanResources.Employee och Person.Person och sortera efter BusinessEntityID.

```
In [156]:
# 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 sortera efter företagsenhets-ID
# 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[156]:

	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
5	6	Jossef	Goldberg	Design Engineer
6	7	Dylan	Miller	Research and Development Manager
7	8	Diane	Margheim	Research and Development Engineer
8	9	Gigi	Matthew	Research and Development Engineer
9	10	Michael	Raheem	Research and Development Manager
10	11	Ovidiu	Cracium	Senior Tool Designer

4-d (Samma fråga som 4A, läser bara data från en annan tabell)

Fråga från HumanResources.vEmployee-vyn, läs BusinessEntityID, FirstName, LastName och JobTitle-informationen och sortera dem i stigande ordning efter anställdas position.

In [157]:
```
# 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), JobTitle (job title)

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 vEmployee_data DataFrame
vEmployee_data = pd.read_sql(query_vEmployee, engine)

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

Out[157]:

	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
5	242	Deborah	Poe	Accounts Receivable Specialist
6	243	Candy	Spoon	Accounts Receivable Specialist
7	244	Bryan	Walton	Accounts Receivable Specialist
8	267	Karen	Berg	Application Specialist
9	268	Ramesh	Meyyappan	Application Specialist
10	269	Dan	Bacon	Application Specialist

4-e Försök att fråga den anställdes lönehistorik från EmployeePayHistory

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_NAME LIKE '%Salary%' OR TABLE_NAME LIKE '%Wage%')
"""

# 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

4-f Försök att fråga den anställdes lönehistorik från 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

4-g Slå samman vEmployee_data och pay_history_data, baserat på det vanliga 'BusinessEntityID', för att få data BusinessEntityID, FirstName, LastName, JobTitle, RateChangeDate, Rate, PayFrequency,

```
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 "BusinessEntityID"
combined_data = pd.merge(vEmployee_data, pay_history_data, on='BusinessEntityID', how='inner')

# 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

4-h e. Frågan väljer data från EmployeePayHistory-tabellen och vEmployeeDepartment-vyn och kopplar samman dessa genom fältet BusinessEntityID.

Tabellen EmployeePayHistory visar den anställdes löneändringsdatum, lönesats och löneutbetalningsfrekvens.

Vyn vEmployeeDepartment visar medarbetarens nummer, namn, efternamn, titel och avdelning.

Frågeresultaten sorteras efter anställds nummer och läses in i DataFrame genom funktionen read_sql i Pandas för att underlätta dataanalys och bearbetning. Detta gör det enkelt att se och analysera anställdas lönestatus och avdelningsrelationer.

```
In [161]:
# 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vEmployeeDepartment视图联合
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 DataFrame
combined_data = pd.read_sql(query, engine)

# 显示查询结果
combined_data
```

Out[161]:

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

5. Extrahera nödvändiga data och gör lönekonfidensintervallanalys

5-a För att få relativt enkla data, försök att bara extrahera och fråga lönenivån på Research and Development

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 DataFrame
combined_data = pd.read_sql(query, engine)

# 显示查询结果
combined_data
```

Out[162]:

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

5-b För testa Efter att ha extraherat och frågat efter lönenivån för Research and Development, beräkna konfidensintervallet för de anställdas inkomster på denna avdelning

In [166]:
```
# För testa 在提取，并查询， Research and Development 部门的工资水平后，计算这个部门的员工收入的置信区间
#För testa extraherar och ifrågasätter lönenivån på forsknings- och utvecklingsavdelningen
#och beräknar sedan konfidensintervallet för inkomsterna för anställda på denna avdelning.

# 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)
```

5-c Extrahera lönedata för alla anställda och beräkna konfidensintervallet för anställdas inkomst på alla avdelningar

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_err)

    # 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)
```