Question 1:

1.B
2.A and B
3.A
4.C
5.C

Question 2:

1- If the attribute “Salary” needs to be discretized into three pay bands, suggest a simple yet sensible solution for the discretization based with a valid argument.

By using 5-Q, defining salary that less than Q2 as “low”,[Q2,Q4) as “medium”, more than Q4 as “High”.

2- Miss Davis's salary is unknown, and the unknown value needs to be imputed, what is a sensible replacement value and why?

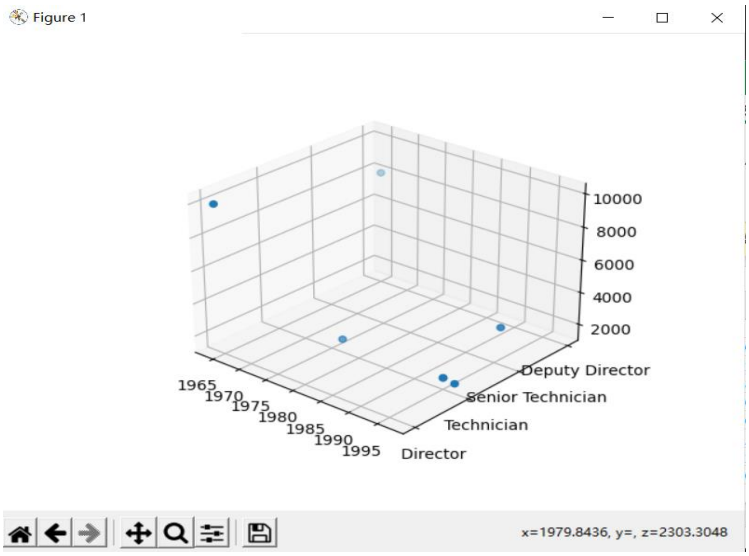
The mean of total salary maybe a reasonable replacement value. If we want to find any information in the salary table, replacing the null value with the average value will not have a big impact on the whole.

3- Among the employee records, which record can be considered as an outlier? What harm can an outlier cause to the understanding of the dataset?

4-

10000 and 8000 may be considered as outliers. Outliers have a great influence on the average, for example, the average salary is 3971 whereas 2020 after removing 10000 and 8000.

4- Draw a diagram of a 3D view using the following attributes: Year of Birth, Status, and Salary.



```
ax.scatter3D(d['Year of Birth'],d[' Status'], d[' Salary'],    cmap='Greens')
ax.set_yticks((0,1,2,3.))
ax.set_yticklabels((' Director ',' Technician ',' Senior Technician ',' Deputy Director '))
plt.show()
```

Convert salary to floats, convert status to (0,1,2,3)

5- What do the data points inside the cube represent? Use the cube as an example to discuss the meaning of OLAP operations such as pivoting, slicing and dicing, rolling up and drilling down.

Data points inside the cube represent base cuboids.

rolling up: Income of employees in the same position (same date of birth)

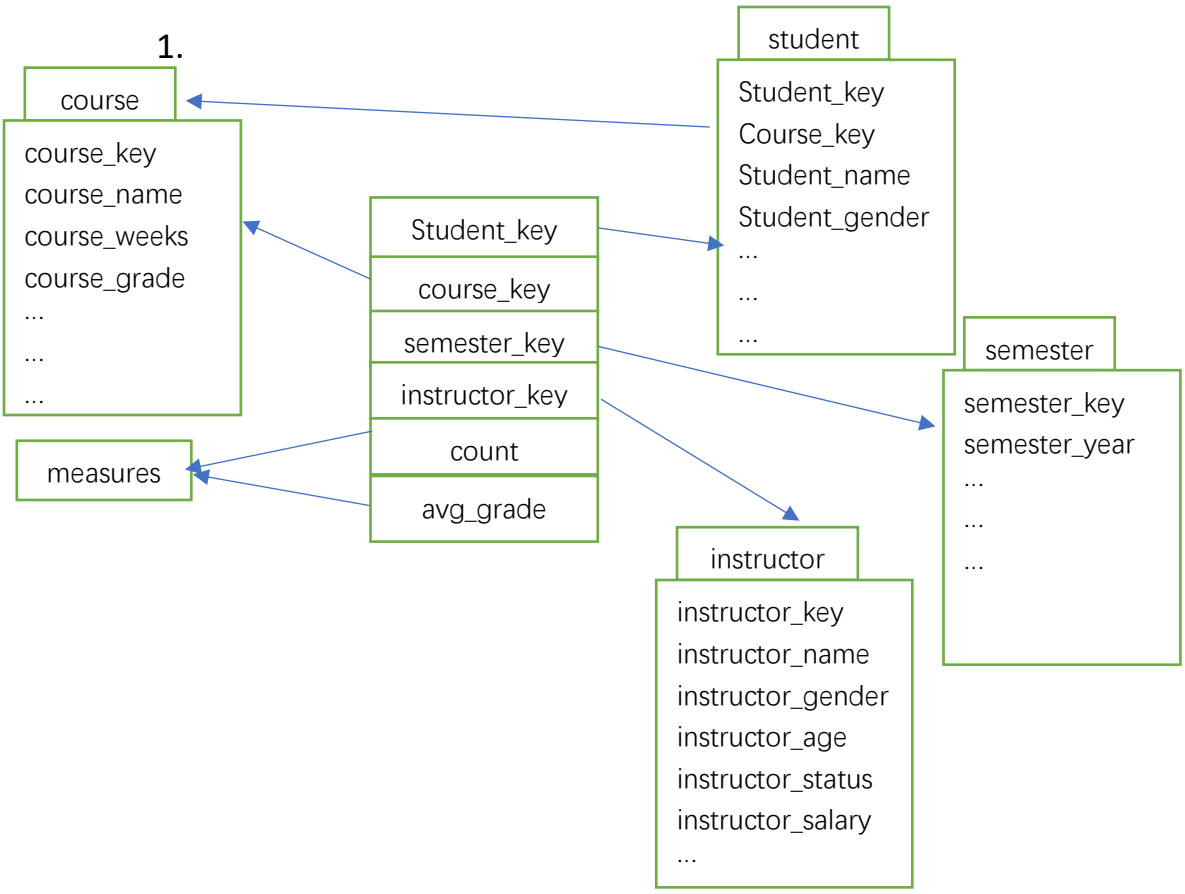
drilling down: From the date of birth to the date of birth and position means a person with the same age to a person with the same position and age.

slicing and dicing: Find a specific data.

Pivoting: View or "perspect" data in various ways, such as the z-axis directly facing the screen direction, you can see the distribution of age and position.

Question 3

1.



2.

roll-up from semester to year, roll-up from course to CS course, Slice and dice the students dimension.

3.

32 cuboids

4.

Create student table with psycopg2:

```
conn = psycopg2.connect(database="testdb", user="postgres", password="8712418", host="127.0.0.1", port="5432")
cur = conn.cursor()
cur.execute('''CREATE TABLE Students
        (StudentNumber INT PRIMARY KEY NOT NULL,
        NAME TEXT NOT NULL,
        AGE INT NOT NULL,
        GENDER CHAR(50) check (GENDER in('Male','Female','Unwilling to disclose'))
        );''')
print ("Table created successfully")

conn.commit()
conn.close()
```

6.

Copy csv files to postgresql with psycopg2:

```
conn = psycopg2.connect(database="testdb", user="postgres", password="8712418", host="127.0.0.1", port="5432")
cur = conn.cursor()
# Import a CSV file into a table using COPY statement
cur.execute('''COPY students(studentnumber,name,age,gender)
FROM 'S:\Data minng\Practical03-idea\specs\input_DW_data.csv'
DELIMITER ','
CSV HEADER;''') # Absolute path
```

```
print ("Table copied successfully")
conn.commit()
conn.close()
```

	studentnumber [PK] integer	name text	age integer	gender character (50)
1	20210848	A	22	Male
2	20213333	D	23	Unwilling to disclose
3	20216678	C	12	Male
4	20218888	B	33	Female
5	20218999	E	36	Male

```
"S:\Data minng\Practical03-idea\venv\Scripts\python.exe" "S:/Data minng/Practical03-idea/main.py"
Table copied successfully
```

7.

Import the sqlalchemy package, create an engine link postgresql, and use sql statements to implement insert, update, and query operations.

```
def read_record (field, name, engine):
    select = "select "+name+" from "+ field
    result = engine.execute(text(select))
    print(result.keys())
    print(result.fetchall())

def update_record (name, details, engine):
    update = "update "+name+" set "+details
    result = engine.execute(text(update))
    print(result.rowcount)

def write_record (field ,name, newvalue, engine):
    insert = "insert into "+field+"("+name+") "+"values("+newvalue+")"
    result = engine.execute(text(insert))
    print(result.rowcount)

def read_dataset (field,engine):
    read = "SELECT * FROM "+field
    result = engine.execute(text(read))
    print(result.fetchall())
    print("read success")
```

```
def write_dataset (name, dataset, engine):
    write = "create table "+name+dataset
    result= engine.execute(text(write))

def list_datasets (engine):
    show = "SELECT *FROM pg_catalog.pg_tables WHERE schemaname != 'pg_catalog' AND schemaname != 'information_schema'"
    result = engine.execute(text(show))
    h = pd.DataFrame(result)
    #Converted to a data frame, so that all table names can be printed, while the fetchall method cannot
    print(h[1])
    print(result.rowcount)
```