QUESTION 01

```
CREATE TABLE warehouse(
        name VARCHAR(20),
        product_id INT,
        units INT,
        PRIMARY KEY(name, product_id, units)
);

CREATE TABLE products(
        product_id INT PRIMARY KEY,
        product_name VARCHAR(20),
        Width INT,
        length INT,
        height INT
);

INSERT INTO warehouse VALUES ('LCHouse1', 1, 1);
INSERT INTO warehouse VALUES ('LCHouse1', 2, 10);
INSERT INTO warehouse VALUES ('LCHouse1', 3, 5);
INSERT INTO warehouse VALUES ('LCHouse2', 1, 2);
INSERT INTO warehouse VALUES ('LCHouse2', 2, 2);
INSERT INTO warehouse VALUES ('LCHouse3', 4, 1);

INSERT INTO products VALUES(1, 'LC-TV', 5, 50, 40);
INSERT INTO products VALUES(2, 'LC-KeyChain', 5, 5, 5);
INSERT INTO products VALUES(3, 'LC-Phone', 2, 10, 10);
INSERT INTO products VALUES(4, 'LC-T-shirt', 4, 10, 20 );

ALTER TABLE products
ADD volume INT;

SELECT
        w.name AS warehouse_name,
        CASE
          WHEN w.name ='LCHouse1' THEN volume = units
*(width*height*length)
          END AS volume
FROM products p
INNER JOIN warehouse w
ON p.product_id = w.product_id
GROUP BY w.name;
```

QUESTION 2

```
CREATE TABLE employee(
	employee_id INT PRIMARY KEY,
	team_id INT
);
INSERT INTO employee(employee_id, team_id) VALUES(1, 8);
INSERT INTO employee(employee_id, team_id) VALUES(2, 8);
INSERT INTO employee(employee_id, team_id) VALUES(3, 8);
INSERT INTO employee(employee_id, team_id) VALUES(4, 7);
INSERT INTO employee(employee_id, team_id) VALUES(5, 9);
INSERT INTO employee(employee_id, team_id) VALUES(6, 9);

SELECT * FROM employee

SELECT employee.employee_id,
	COUNT(CASE WHEN team_id = 8 THEN SUM(team_id =) END)
FROM employee
GROUP BY employee_id;

SELECT employee_id,
	CASE
		WHEN team_id = 8 THEN (Select Count(*) from employee where
team_id = 8)
		WHEN team_id = 7 THEN (Select Count(*) from employee where
team_id = 7)
		WHEN team_id = 9 THEN (Select Count(*) from employee where
team_id = 9)
		END AS team_size
FROM employee
GROUP BY employee_id;
```

OUTPUT:

	employee_id [PK] integer	team_size bigint
1	2	3
2	5	2
3	4	1
4	6	2
5	3	3
6	1	3

QUESTION 4

```
CREATE TABLE salaries(
        company_id INT,
        employee_id INT,
        employee_name VARCHAR(20),
        salary INT,
        PRIMARY KEY (company_id, employee_id)
);

INSERT INTO salaries VALUES(1, 1, 'Tony', 2000);
INSERT INTO salaries VALUES(1, 2, 'Pronub', 21300);
INSERT INTO salaries VALUES(1, 3, 'Tyrrox', 10800);
INSERT INTO salaries VALUES(2, 1, 'Pam', 300);
INSERT INTO salaries VALUES(2, 7, 'Bassem', 450);
INSERT INTO salaries VALUES(2, 9, 'Hermione', 700);
INSERT INTO salaries VALUES(3, 7, 'Bpcaben', 100);
INSERT INTO salaries VALUES(3, 2, 'Ognjen', 2200);
INSERT INTO salaries VALUES(3, 13, 'Nyancat', 3300);
INSERT INTO salaries VALUES(3, 15, 'Morninngcat', 7777);

SELECT * FROM salaries;

SELECT
company_id,
employee_id,
employee_name,
salary AS salary_after_tax,
        CASE
                WHEN  (ROUND (MAX(salary)>10000 OVER(PARTITION BY
s.company_id)), 0)
                THEN salary_after_tax = salary - (24/ 100) * salary
                WHEN  (ROUND (MAX(salary)>10000 OVER(PARTITION BY
s.company_id)), 0)
                THEN salary_after_tax = salary - (24/ 100) * salary
                END
FROM salaries
GROUP BY company_id;
```

QUESTION 5 MAP-REDUCE

----------mapper function----------

```
import sys
def mapper(line):
keypair[]
ip_words = line.split()          #splitting line into words
for word in ip_words:
        keypair.append([word, 1])
return keypair                   #returning he key-pair value from the mapper
```

---------- reducer function----------

```
def reducer(keypair_list):
result = {}
for i in keypair_list:
        for j in i:
                if j[0] in result:
                        result [j[0]] += j[1]
                else:
                        resut[j[0]] = j[1]
return result;
```

```
def input_splits():
ip_para = takeinput()                          #taking input
mapper_result = []
for line in ip_para:
        mapper_result.append(mapper(line))              #calling mapper and
                                                appending result in mapper_result
reducer_result = resucer(mapper_result)
print (reducer_result)                                  #final result
```

```
#take multiline input
```

```
def takeInput():
print(“enter your para”)
input = sys.stdin.readlines()
return input
input_splits()
```