XJTLU, Information and Computing Science
CPT 103 Semester 1

Report of DurianPC Company Database Design

Pin Qian

November 2020

Contents

1. Purpose 2
2. Hypothesis 2
2.1 Company related hypothesis 2
2.2 Data related hypothesis 2
3. Discussion 2
3.1 Requirement Analysis and Design Logic 2
3.2 Usage Guideline 12
4. Conclusion 16
Appendix 16
INSERT 16
SELECT 20

1. Purpose

This report is aiming to present the design logic and its structure is as follows. Firstly, hypothesis will be presented to make sure the rationality and integrity. Secondly, in discussion part, “CREATE” related work will be delivered in section3.1, while “INSERT” and “SELECT” related work will be shown in section3.2. Finally, a conclusion will be given to summarize.

2. Hypothesis

2.1 Company related hypothesis

- Since staff ID number is an important attribute which will be used as reference in many other relations, so it is reasonable to assume that each staff has a unique ID number, and the ID number type will not be changed.

- Since the requirement handbook only give an example of southern region, here I assume that the formats of postcode are “LSxxxxx” in South, “LNxxxxx” in North, “LCxxxxx” in the central of Lukewarm Kingdom.

- For each postcode, it can be used as up to 1 same type workplace. For example, a place with postcode LN00001 can have a factory and a retail store, but there will not be two retail stores simultaneously.

- As for evaluation, I assume whether a staff was a high performer will be evaluated once a year, and whether a staff got sick or worked in south region once a month, since staff may change workplace within a year and get sick many times in a year.

2.2 Data related hypothesis

- Besides specific requirements mentioned in the task sheet, all string data is regulated in varchar(255), integer data is regulated in int(11) for convenience.

3. Discussion

3.1 Requirement Analysis and Design Logic

- **What information to save**

  This is a database designed for a company. Generally, the requirement can be classified into 2 modules - company related module, staff related module.

  In company related module, for structure aspect, three **entities** – *workplace*, *office*,

department are selected. Since the task sheet need to know the relation between office and department, which is a m:m relation, so an *office_department* table is created. For business aspect, *computer, component, manufacturer* is selected.

In staff related module, *staff*, *role*, *email, evaluation* and *salary* are selected. In addition, two bridge table – *staff_role*, *staff_email* are designed due to m:m relation.

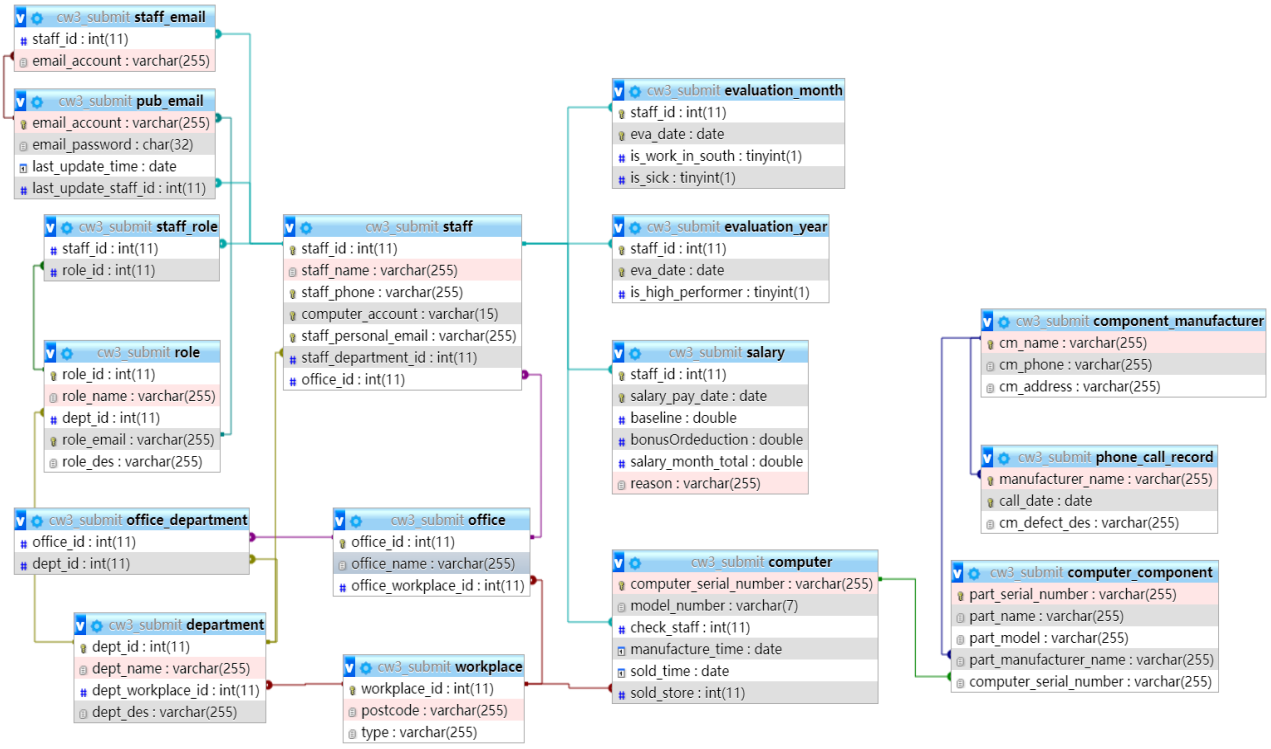
The reason for bridge table and the relation between entity will be explained later.

- **What functions to realize**

  According to the requirement, 12 main functions should to be realized, which will be presented in the Usage Guideline section.

3.1.1 ER diagram

While doing the coursework, I first selected the main entities and identified their relations, and then listed attributes. However, since we can't have m:m relation and must be in 3NF, I modified the database several times and after carefully check, finally, the ER diagram is presented below.



- **Creating procedure and relation declaration**

  1. Start from *workplace*. It is the first table I created, since later *department* and *office* need the {workplace_id} as reference to constrain their workplace. A workplace has many offices and departments, so the relations are both 1: m.

  2. Then *department* and *office* were created, which are both staff's reference. An office has many staffs, so the relation office to staff is 1: m. Meanwhile, {*role_id*} references

to *department*'s {dept_id}, in which the relation between role and department is 1: m since s department can have many roles, but one role decided by name and department only belong to one department. In addition, *office_department* is created since an office may have different departments, and a department has many offices.

3. Now we have department, so now create *role,* whose {role_email} is a reference for *pub_email*'s {email_account}. A role has a shared email, so the relation is 1: 1.

4. Then *pub_email* and the most important table *staff* was created. As the bule line shows, {staff_id} is a reference by many attributes in other tables. Among them, there are several m: m relations. A role can be allocated to many staffs, a staff can have many roles, so the relation is m:m, thus *staff_role* table is created; One staff may have many role emails, and one role email can also be used by many staffs, thus *staff_email* is created; For role shared email, it can be used by many staffs, and one staff may have many role emails, so *staff_email* is created.

5. Then *evaluations* and *salary* are created, whose {staff_id} reference to staff's {staff_id}. A staff has many evaluation records and salary records, so the relation is 1: m.

6. Then *component_manufacturer* is created, whose {cm_name} will be used as reference. Then, *computer* is created, whose {sold store} reference to *workplace*'s {workplace_id}. A computer only can be sold is a store, so the relation computer to workplace is 1: 1; A computer has many components, so the relation is 1: m.

7. Finally, *computer_component* and *phone_call_record* are created before *computer* and *manufacturer* since attributes of them will be used as references. A component can be produced by different manufacturer, so the relation is 1: m.

In the next section, detailed table design will be presented.

3.1.2 Table Design

- **Workplace**

Attributes	Type	Constrains	Notes
workplace_id	int(11)	PK	
postcode	varchar(255)	NOT NULL	Locates the region.
type	varchar(255)	NOT NULL	Distinguishes whether the workplace is HQ, factory or retail store

Functional Dependencies
{workplace_id} → {postcode, type}

Primary Key

Although each region is associated with a unique postcode, a possible situation is that there may be both factory and retail store in one region. In that case, given a postcode of a workplace, we can't tell its type. Hence, {postcode} can't be the primary key. The candidate key {postcode, type} seems to be a good choice, but I finally decide to use a surrogate key {workplace_id} as primary key for further convenience.

- **Office**

Attributes	Type	Constrains	Notes
office_id	int(11)	PK	
office_name	varchar(255)	NOT NULL	Locates the room location in that workplace.
office_workplace_id	int(11)	FK	Identify the office belong to which workplace. Reference to workplace(workplace_id)

Functional Dependencies

{office_id} → {office_name, office_workplace_id}

Reason for Primary Key

{office_id} is chosen to be the primary key.

The reason for not choosing {office_name} or {office_name, office_workplace_id} is that offices in different workplaces may have the same name, and a composite primary key may cause inconvenience for further change.

Reason for Foreign Key

Since only if the workplace exists, can an office in that workplace exists, therefore, a foreign key constrain is required for {office_workplace_id}.

- Department

Attributes	Type	Constrains	Notes
dept_id	int(11)	PK	
dept_name	varchar(255)	NOT NULL	Identify the department's name in that workplace.
dept_workplace_id	int(11)	FK	Identify the department belong to which workplace. Reference to workplace(workplace_id).
dept_des	varchar(255)		

Functional Dependencies
{dept_id} → {dept_name, dept_workplace_id, dept_des}

Reason for Primary Key
{dept_id} is chosen to be the primary key.
The reason for not choosing {dept_name} or {dept_name, dept_workplace_id} is similar to the explanation of office table.

Reason for Foreign Key
Since only if the workplace exists, can a department in that workplace exists, therefore, a foreign key constrain is required for {dept_workplace_id}.

- office_department

Attributes	Type	Constrains	Notes
office_id	int(11)	FK	Reference to office(office_id)
dept_id	Int(11)	FK	Reference to department(dept_id)

Offices are usually allocated to staffs by the department, but staffs from different departments can occasionally share offices. That means one department has many offices while one office also may have many departments. Thus, we need this bridge table to solve the m:m problem.

Reason for Foreign Key
Only if the office exists, can a department have that office, similarly, only if the department exists, can an office be allocated to that department. Thus, these two foreign keys are required.

- Staff

Attributes	Type	Constrains	Notes
staff_id	int(11)	PK	
staff_name	varchar(255)	NOT NULL	
staff_phone	varchar(255)	UNIQUE	
computer_account	varchar(15)	UNIQUE	
staff_personal_email	varchar(255)	UNIQUE	
staff_department_id	int(11)	FK	Reference to department(dept_id)
office_id	int(11)	FK	Reference to office(office_id)

Functional Dependencies
{staff_id} → {staff_name, staff_phone, computer_account, staff_personal_email, staff_department_id, office_id}
PS: It seems that we have the functional dependency of {staff_id} →{staff_department_id} →{office_id}, but it is not true. Since the relation between department and office is in the *office_department* table. In this table, the department and office don't have the functional dependency, they each only have functional dependency on {staff_id}.

Reason for Primary Key
Since each staff has his unique ID.

Reason for Unique Key
Since each staff has a unique phone number, computer account, personal email, so it is necessary to constrain those attributes a unique key.

Reason for Foreign Key
Only existing department and office can be used in this table.

- Role

Attributes	Type	Constrains	Notes
role_id	int(11)	PK	
role_name	varchar(255)	NOT NULL	
dept_id	Int(11)	FK	Reference to department(dept_id)
role_email	varchar(15)	UNIQUE	Means that an email will be shared within a same role
role_des	varchar(255)		

Functional Dependencies
{role_id} → {role_name, dept_id, role_email, role_des}

Reason for Primary Key
Since different departments' role may have a same name, {role_id} is chosen to be the primary key. In this way, it will be more convenient to reference this attribute than using {role_name, dept_id} as composite primary key.

Reason for Unique Key
According to the requirement handbook, each role only has one public email.

Reason for Foreign Key
Only existing department can be used in this table.

- staff_role

Attributes	Type	Constrains	Notes
staff_id	int(11)	FK	Reference to staff(staff_id)
role_id	Int(11)	FK	Reference to role(role_id)

Since knowing that one staff can have different roles, while one role can be assigned to different staffs, so it is a m:m relation. Therefore, a bridge table is needed.

Reason for Foreign Key:
Since only if the staff exists, can a role be assigned to that staff, similarly, only if the role exists, can a staff have that role. Thus, they are required.

- pub_email

Attributes	Type	Constrains	Notes
email_account	varchar(255)	PK,FK	Reference to role(role_email)
email_password	char(32)		
last_update_time	date		
last_update_staff_id	varchar(255)	FK	Records the last updating staff Reference to staff(staff_id)

Functional Dependencies
{email_account} → {email_password, last_update_time, last_update_staff_id}

Reason for Primary Key
Each email account is unique, there is no need to use other attributes.

Reason for Foreign Key
For {email_account}, only occurred in "role_email" in role table can be added into this table. For {last_update_staff_id}, only existing staff can be added into this table.

- staff_email

Attributes	Type	Constrains	Notes
staff_id	int(11)	FK	Reference to staff(staff_id)
email_account	varchar(255)	FK	Reference to role(role_email)

Since knowing that one staff can have more than one emails, while one email can be shared with different staffs, so it is a m:m relation. Therefore, a bridge table is needed.

Reason for Foreign Key
Since only if the staff exists, can an email be assigned to that staff, similarly, only if the email exists, can a staff have that email. Thus, these two foreign keys are required.

- evaluation year

Attributes	Type	Constrains
staff_id	int(11)	PK
eva_date	varchar(255)	
is_high_performer	boolean	

- evaluation month

Attributes	Type	Constrains
staff_id	int(11)	PK
eva_date	varchar(255)	
is_work_in_south	boolean	
is_sick	boolean	

The evaluation mainly comes from three aspects.
1. Company will judge whether a staff is high performer once a year.
2. Whether a staff got sick will be recorded once a month.
3. Considering that staff may change workplace, so whether working in south will be recorded once a month.

Reason for Primary Key:
In these two tables, {staff_id, eva_date} is the primary key. Since every year(month)'s evaluation of one staff will be kept in database, so {staff_id, eva_date} satisfy uniqueness and minimality.

Functional Dependencies:
{staff_id, eva_date} → {is_high_performer}

{staff_id, eva_date} → {is_work_in_south, is_sick}

- Salary

Attributes	Type	Constrains
staff_id	int(11)	PK
salary_pay_date	varchar(255)	PK
baseline	double	

bonusOrdeduction	double	
salary_month_total	double	
reason	varchar(255)	

Functional Dependencies
{staff_id, salary_pay_date} -> {baseline, bonusOrdeduction, salary_month_total, reason}

Reason for Primary Key
{staff_id, salary_pay_date} is the primary key. Since every month payment of one staff will be kept in database, so {staff_id, salary_pay_date} satisfy uniqueness and minimality.

Reason for Foreign Key
The reason is similar to previous description, to avoid redundancy, I will not repeat here.

- component_manufacturer

Attributes	Type	Constrains
cm_name	varchar(255)	PK
cm_phone	varchar(255)	NOT NULL
cm_address	varchar(255)	NOT NULL

Functional Dependencies
{cm_name} → {cm_phone, cm_address}

Reason for Primary Key
Manufacturer's name is unique due to the principles of factory registration.

- phone_call_record

Attributes	Type	Constrains
manufacturer_name	varchar(255)	PK, FK
call_date	date	PK
cm_defect_des	varchar(255)	

This table is designed for recording every phone call to manufacturer. The call_date can be null because sometimes there may not be necessary to make a call in that month.

Functional Dependencies:
{manufacturer_name, call_date} → {effect_des}

Reason for Primary Key and Foreign Key:

Manufacturer's name and call date can decide a record.
Manufacturer must exist, so foreign key constrain is needed.

- Computer

Attributes	Type	Constrains	Notes
computer_serial_number	varchar(255)	PK	
model_number	date		
check_staff	int(11)	FK	Reference to staff(staff_id)
manufacture_time	date		
sold_time	date		If null, shows that it have not been sold.
sold_store	int(11)	FK	Reference to workplace(workplace_id)

Functional Dependencies
{computer_serial_number}→ {model_number, check_staff, manufacture_time, sold_time, sold_store}

Reason for Primary Key
Each computer has a unique serial number.

Reason for Foreign Key
Staff and the retail store must exist, so foreign key constrain is needed.

- computer_component

Attributes	Type	Constrains	Notes
part_serial_number	varchar(255)	PK	
part_name	varchar(255)		
part_manufacturer_name	varchar(255)	FK	Reference to component_manufacturer(cm_name)
computer_serial_number	varchar(255)	FK	Reference to computer(computer_serial_number)

Functional Dependencies
{part_serial_number} → {part_name, part_manufacturer_name, computer_serial_number}

Reason for Primary Key
Each component of computer has a unique serial number.

Reason for Foreign Key
Manufacturer and the computer serial number must exist, so foreign key constrain is needed.

3.2 Usage Guideline

In this section, “SELECT” related sentence will be presented. In order to have an easy and enjoyable experience, some example data will be created.

3.2.1 Data shaping
The data I used for “select” part is provided in Appendix.

3.2.2 Main Function Operation
In this section, 12 functions and its SQL code will be presented. I will give example to show how to achieve the functions.

- Given a department, list all staffs in that department
  e.g. To know who is in the sales department of LN00001 retail store.

```
SELECT staff.staff_id AS ID,staff.staff_name AS Name FROM staff
WHERE staff.staff_department_id =
(SELECT department.dept_id FROM department
WHERE department.dept_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
 AND department.dept_name = "sales dept.");
```

The result is

ID	Name
1928149	staff1
1928150	staff2

Generally, only need to change the data in red can search other’s department.

- Given an office, list all staffs in that office
  e.g. To know who is working at Room02, LN00001.

```
SELECT staff.staff_id AS ID,staff.staff_name AS Name FROM staff
WHERE staff.office_id =
(SELECT office.office_id FROM office
WHERE office.office_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
AND office.office_name = "Room02");
```

The result is

staff_id
1928149
1928150

Generally, only need to change the data in red can search other's office.

- Given an office, list all departments in that office
  e.g. To know what departments are in the Room03 office at a retail store with LN00001

```
SELECT staff.staff_id AS ID,staff.staff_name AS Name FROM staff
WHERE staff.office_id =
(SELECT office.office_id FROM office
WHERE office.office_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
AND office.office_name = "Room02");
```

The result is

DepartmentID
12
13

Generally, only need to change the data in red can search other's office.

- Given a staff, list all roles the staff has
  e.g. To know which roles the employee with ID 1928149 is responsible for

```
SELECT staff_role.role_id AS RoleID FROM staff_role
WHERE staff_role.staff_id = 1928149;
```

The result is

RoleID
1
2

Generally, only need to change the data in red can search other staff's roles.

- Given a role, list all staffs related to the role.
  e.g. To know which staffs are responsible for the role of role1 in the sales department of a retail store with LN00001

```
SELECT staff_role.staff_id from staff_role
where staff_role.role_id =
(SELECT role.role_id FROM role
WHERE role.role_name = "role1" and role.dept_id =
(SELECT department.dept_id FROM department
WHERE department.dept_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
 AND department.dept_name = "sales dept."));
```

The result is

staff_id
1928149
1928150

Generally, only need to change the data in red can search other role's staffs.

- Given a role, list the shared email.
  e.g. To know the email corresponding to the role named Role1 in the sales department of LN00001 retail store.

```
SELECT role.role_id AS RoleID, role.role_name AS RoleName,
role.dept_id as DepartmentID, role.role_email AS "Shared Email" FROM role
WHERE role.role_id =
(SELECT role.role_id FROM role
WHERE role.role_name = "role1" and role.dept_id =
(SELECT department.dept_id FROM department
WHERE department.dept_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
 AND department.dept_name = "sales dept."));
```

The result is

RoleID	RoleName	DepartmentID	Shared Email
1	role1	13	role1@qq.com

Generally, only need to change the data in red.

- Given a shared email, list the department the email belongs to
  e.g. To know which department role1@qq.com belongs to

```
SELECT department.dept_id AS DeptID,
department.dept_name AS deptName,role.role_email AS Email FROM department,role
WHERE department.dept_id =
(SELECT role.dept_id FROM role
WHERE role.role_email = "role1@qq.com") AND role.role_email = "role1@qq.com";
```

The result is

DeptID	deptName	Email
13	sales dept.	role1@qq.com

- Given a shared email, list all the staffs who can use it
  e.g. To know who can use role1@qq.com

```
SELECT staff_role.staff_id AS StaffID FROM staff_role
WHERE staff_role.role_id =
(SELECT role.role_id FROM role
WHERE role.role_email = "role1@qq.com");
```

The result is

StaffID
1928149
1928150

- Given a shared email, list how many valid days left

e.g. role3@qq.com

```
SELECT
DATEDIFF(DATE_ADD(pub_email.last_update_time,
INTERVAL 1 YEAR),pub_email.last_update_time)
AS "ValidRemaining Days" FROM pub_email
WHERE pub_email.email_account =
"role3@qq.com";
```

The result is

ValidRemaining Days
365

- Given time and staff ID, compute the salary based on perform
  e.g. To compute 1928151 staff's salary in 2019-10

```
SELECT salary.staff_id AS StaffID,
evaluation_month.eva_date AS Date,
evaluation_year.is_high_performer AS "If High Performer",
evaluation_month.is_work_in_south AS "If Working in South",
evaluation_month.is_sick AS "If Sick",
salary.baseline AS baseline,
salary.baseline*(1+0.2*evaluation_year.is_high_performer+0.1*evaluation_month.is_work_in_south)*(1-
0.1*evaluation_month.is_sick) AS Salary,
(salary.baseline*(1+0.2*evaluation_year.is_high_performer+0.1*evaluation_month.is_work_in_south)*(1-
0.1*evaluation_month.is_sick))- salary.baseline AS BounsOrDeduction FROM(
(salary JOIN evaluation_year ON evaluation_year.staff_id = salary.staff_id) JOIN evaluation_month ON
evaluation_month.staff_id = salary.staff_id)
WHERE salary.staff_id = 1928151 AND evaluation_year.eva_date = '2018-12-31' AND evaluation_month.eva_date =
'2019-10-31';
```

The result is

StaffID	Date	If High Performer	If Working in South	If Sick	baseline	Salary	BounsOrDeduction
1928151	2019-10-31	0	1	1	3000	2970.0000000000005	-29.999999999999545

Generally, the result is based on staff ID, the last year, and the month in current year. Only need to change the staff_id and the red information in the last line.

- Give staff ID, list all salary records
  e.g. 1928149 staff's salary records

```
SELECT * FROM salary
WHERE salary.staff_id = 1928149;
```

The result is

staff_id	salary_pay_date	baseline	bonusOrdeduction	salary_month_total	reason
1928149	0000-00-00	4000	*NULL*	*NULL*	*NULL*

Generally, only need to change the data in green.

- Given model number, tell computer's type

```
 #list all desktops。
SELECT * FROM computer
WHERE computer.model_number LIKE "DT%";

#laptop-powerhouse
SELECT * FROM computer
WHERE computer.model_number LIKE "LC%P";

#laptop-s feather-light
SELECT * FROM computer
WHERE computer.model_number LIKE "LC%L";
```

4. Conclusion

To sum up, although in the process of completing this report, I found that there are areas that could be improved better, however, so far, the database I designed can complete the tasks mentioned in the requirements and it has no m:m relationship and all tables are in 3NF.

Appendix

INSERT

workplace
The factory is only in the north, the headquarters is only in the center, and there are retail stores everywhere
INSERT INTO workplace(postcode,type) VALUES
("LC00001", "headquarters"),
("LC00002", "retail store");

INSERT INTO workplace(postcode,type) VALUES
("LN00001", "factory"),
("LN00001", "retail store");

INSERT INTO workplace(postcode,type) VALUES
("LS00001", "retail store"),
("LS00002", "retail store");

department
HQ

```
INSERT INTO department(dept_name,dept_workplace_id,dept_des) VALUES
("HR dept.", 1, NULL),
("Finance dept.", 1, NULL),
("sales dept.", 1, NULL),
("Technical dept.", 1, NULL);

# RS LC00002
INSERT INTO department(dept_name,dept_workplace_id,dept_des) VALUES
("HR dept.", 2, NULL),
("Finance dept.", 2, NULL),
("sales dept.", 2, NULL);

# F LN00001
INSERT INTO department(dept_name,dept_workplace_id,dept_des) VALUES
("HR dept.", 3, NULL),
("Finance dept.", 3, NULL),
("Technical dept.", 3, NULL);

# RS LN00001
INSERT INTO department(dept_name,dept_workplace_id,dept_des) VALUES
("HR dept.", 4, NULL),
("Finance dept.", 4, NULL),
("sales dept.", 4, NULL);

# RS LS00001
INSERT INTO department(dept_name,dept_workplace_id,dept_des) VALUES
("HR dept.", 5, NULL),
("Finance dept.", 5, NULL),
("sales dept.", 5, NULL);

# RS LS00002
INSERT INTO department(dept_name,dept_workplace_id,dept_des) VALUES
("HR dept.", 6, NULL),
("Finance dept.", 6, NULL),
("sales dept.", 6, NULL);

# office
INSERT INTO office(office_name,office_workplace_id) VALUES
("Room01", "4"),
("Room02", "4"),
("Room03", "4");
INSERT INTO office(office_name,office_workplace_id) VALUES
("Room01", "5"),
("Room02", "5"),
```

(''Room03'', ''5'');

office_department
#LN00001 RS
INSERT INTO office_department(office_id,dept_id) VALUES
(1,11),
(2,13),
(3,12),
(3,13);

#LS00001 RS
INSERT INTO office_department(office_id,dept_id) VALUES
(4,14),
(5,16),
(6,15),
(6,16);

staff
INSERT INTO staff(staff_id,staff_name,staff_phone,computer_account,staff_personal_email,staff_department_id,office_id) VALUES
(1928149,''staff1'',''18368009640'',''staff1_ac'',''staff1@qq.com'',13,2),
(1928150,''staff2'',''18368009641'',''staff2_ac'',''staff2@qq.com'',13,2),
(1928151,''staff3'',''18368009642'',''staff3_ac'',''staff3@qq.com'',16,6);

role
INSERT INTO role(role_name,dept_id,role_email,role_des) VALUES
(''role1'',13,''role1@qq.com'', NULL),
(''role2'',13,''role2@qq.com'', NULL),
(''role3'',16,''role3@qq.com'', NULL);

staff_role
INSERT INTO staff_role(staff_id,role_id) VALUES
(1928149,1),
(1928149,2),
(1928150,1),
(1928151,3);

#pub_email
INSERT INTO pub_email(email_account,last_update_time,last_update_staff_id) VALUES
(''role1@qq.com'','2020-02-01',1928149),
(''role2@qq.com'','2020-02-05',1928150),
(''role3@qq.com'','202#0-03-08',1928151);

```
# staff_email
INSERT INTO staff_email(staff_id,email_account) VALUES
(1928149,"role1@qq.com"),
(1928150,"role2@qq.com"),
(1928151,"role3@qq.com");

# evaluation_year
INSERT INTO evaluation_year(staff_id,eva_date,is_high_performer) VALUES
(1928149,'2018-12-31',false),
(1928150,'2018-12-31',false),
(1928151,'2018-12-31',false),
(1928149,'2019-12-31',true),
(1928150,'2019-12-31',false),
(1928151,'2019-12-31',false);

# evaluation_month
INSERT INTO evaluation_month(staff_id,eva_date,is_work_in_south,is_sick) VALUES
(1928149,'2019-9-30',false,true),
(1928150,'2019-9-30',false,false),
(1928151,'2019-9-30',true,false),
(1928149,'2019-10-31',false,true),
(1928150,'2019-10-31',false,true),
(1928151,'2019-10-31',true,true);

# salary
INSERT INTO salary(staff_id,baseline) VALUES
(1928149,4000),
(1928150,5000),
(1928151,3000);

# component_manufacturer
INSERT INTO component_manufacturer(cm_name,cm_phone) VALUES
("cm_name1","18868009640"),
("cm_name2","18868009641"),
("cm_name3","18868009642");

# phone_call_record
INSERT INTO phone_call_record(manufacturer_name,call_date,cm_defect_des) VALUES
("cm_name1",'2020-10-31',"need improve!"),
("cm_name2",'2020-10-31',"good!"),
("cm_name3",'2020-10-31',"bad!");

# computer
INSERT                                                                INTO
```

```
computer(computer_serial_number,model_number,check_staff,manufacture_time,sold_time,sold_store) VALUES
("serial number1","DT4271",1928149,'2020-03-02','2020-03-29',2),
("serial number2","LC2333P",1928150,'2020-01-02','2020-03-29',2);

# computer_component
INSERT INTO computer_component(part_serial_number,part_name,part_model,part_manufacturer_name,computer_serial_number) VALUES
("L070Q228","CPU","i5-4430","cm_name1","serial number1");
```

SELECT

```
# 1
SELECT staff.staff_id AS ID,staff.staff_name AS Name FROM staff
WHERE staff.staff_department_id =
(SELECT department.dept_id FROM department
WHERE department.dept_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
  AND department.dept_name = "sales dept.");

# 2
SELECT staff.staff_id AS ID,staff.staff_name AS Name FROM staff
WHERE staff.office_id =
(SELECT office.office_id FROM office
WHERE office.office_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
AND office.office_name = "Room02");

# 3
SELECT office_department.dept_id AS DepartmentID FROM office_department
WHERE office_department.office_id =
(SELECT office.office_id FROM office
WHERE office.office_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
AND office.office_name = "Room03");


# 4
SELECT staff_role.role_id AS RoleID FROM staff_role
```

```
WHERE staff_role.staff_id = 1928149;

# 5
SELECT staff_role.staff_id from staff_role
where staff_role.role_id =
(SELECT role.role_id FROM role
WHERE role.role_name = "role1" and role.dept_id =
(SELECT department.dept_id FROM department
WHERE department.dept_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
  AND department.dept_name = "sales dept."));

# 6
SELECT role.role_id AS RoleID, role.role_name AS RoleName,
role.dept_id as DepartmentID, role.role_email AS "Shared Email" FROM role
WHERE role.role_id =
(SELECT role.role_id FROM role
WHERE role.role_name = "role1" and role.dept_id =
(SELECT department.dept_id FROM department
WHERE department.dept_workplace_id =
(SELECT workplace.workplace_id FROM workplace
WHERE workplace.postcode = "LN00001" and workplace.type = "retail store")
  AND department.dept_name = "sales dept."));

# 7
SELECT department.dept_id AS DeptID,
department.dept_name AS deptName,role.role_email AS Email FROM department,role
WHERE department.dept_id =
(SELECT role.dept_id FROM role
WHERE role.role_email = "role1@qq.com") AND role.role_email = "role1@qq.com";

# 8
SELECT staff_role.staff_id AS StaffID FROM staff_role
WHERE staff_role.role_id =
(SELECT role.role_id FROM role
WHERE role.role_email = "role1@qq.com");

# 9
SELECT DATEDIFF(DATE_ADD(pub_email.last_update_time,INTERVAL 1
YEAR),pub_email.last_update_time) AS "ValidRemaining Days" FROM pub_email
WHERE pub_email.email_account = "role3@qq.com";

# 10
```

```
SELECT salary.staff_id AS StaffID,
evaluation_month.eva_date AS Date,
evaluation_year.is_high_performer AS "If High Performer",
evaluation_month.is_work_in_south AS "If Working in South",
evaluation_month.is_sick AS "If Sick",
salary.baseline AS baseline,
salary.baseline*(1+0.2*evaluation_year.is_high_performer+0.1*evaluation_month.is_work_in_
south)*(1-0.1*evaluation_month.is_sick) AS Salary,
(salary.baseline*(1+0.2*evaluation_year.is_high_performer+0.1*evaluation_month.is_work_in
_south)*(1-0.1*evaluation_month.is_sick))- salary.baseline AS BounsOrDeduction FROM(
(salary JOIN evaluation_year ON evaluation_year.staff_id = salary.staff_id) JOIN
evaluation_month ON evaluation_month.staff_id = salary.staff_id)
WHERE salary.staff_id = 1928151 AND evaluation_year.eva_date = '2018-12-31' AND
evaluation_month.eva_date = '2019-10-31';

# 11
SELECT * FROM salary
WHERE salary.staff_id = 1928149;

#12
## list all desktops。
SELECT * FROM computer
WHERE computer.model_number LIKE "DT%";

## laptop-powerhouse
SELECT * FROM computer
WHERE computer.model_number LIKE "LC%P";

## laptop-s feather-light
SELECT * FROM computer
WHERE computer.model_number LIKE "LC%L";
```