Database Systems Term Project

CS 4347

# 

# 

# Group 8

Jonathan Haerr

Shrey Saraswat

Rhed Santiago

# Table of Contents

[**Problem Statement**](#_2yhv2cx03s2m) **3**

[**Individual Contributions**](#_qekqj7k0745v) **3**

[**EER Diagram**](#_8yyg9buf0rec) **4**

[**Assumptions**](#_ej43gvdf6oav) **4**

[**DB Schema**](#_v7jbm2wpckdn) **5**

[**Normalization Process**](#_bpb6bnjxtnmg) **7**

[First Normal Form](#_gsccagn7z7iu) 7

[Second Normal Form](#_m733nk95k27d) 7

[Third Normal Form](#_100z61dkf306) 7

[Dependency Diagrams](#_5ll52bkszh92) **8**

[**MySQL Implementation**](#_ntxk1owx29c0) **10**

[Create Database](#_c9kmlyxn61hn) 10

[Creating Views](#_fa2puqg60p44) 13

[Answering Queries](#_v5qb97getrdq) 13

# Problem Statement

The purpose of this project is to design a database system for ABC company. We start by developing an entity-relationship diagram that defines a company’s individual entities, their attributes, and the relationships that connect the entities. Using the EER diagram, we then map the database into a relational database schema. We normalize the relations in this schema by creating dependency diagrams. Once this framework is established, we implement the database design using MySQL and perform a set of sample queries to demonstrate the integrity of our database system.

# 

# Individual Contributions

The breakdown for the contribution of the project is as follows:

Shrey:

* Helped create EER diagram
* Created and normalized dependency diagrams
* Helped edit code and write queries
* Helped write the report

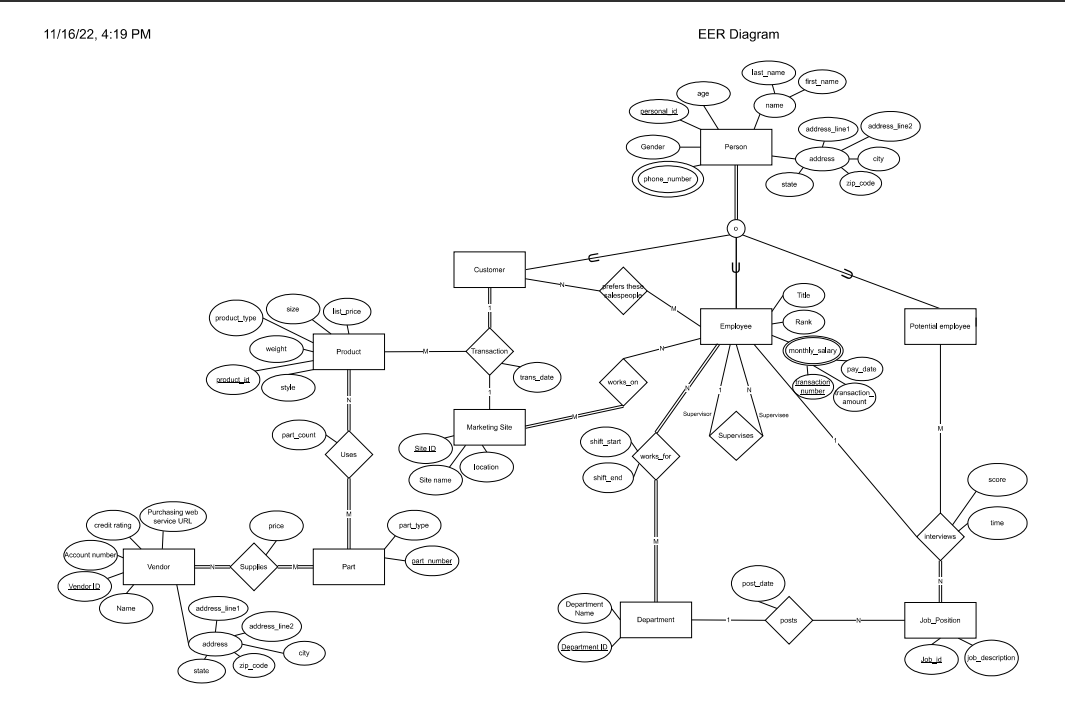
Jonathan:

* Helped create EER diagram
* Helped edit DB schema and dependencies
* Implemented database in MySQL; developed views and queries
* Helped write the report

Rhed:

* Helped create EER diagram
* Developed DB schema
* Helped edit code and write queries
* Helped write the report

# EER Diagram

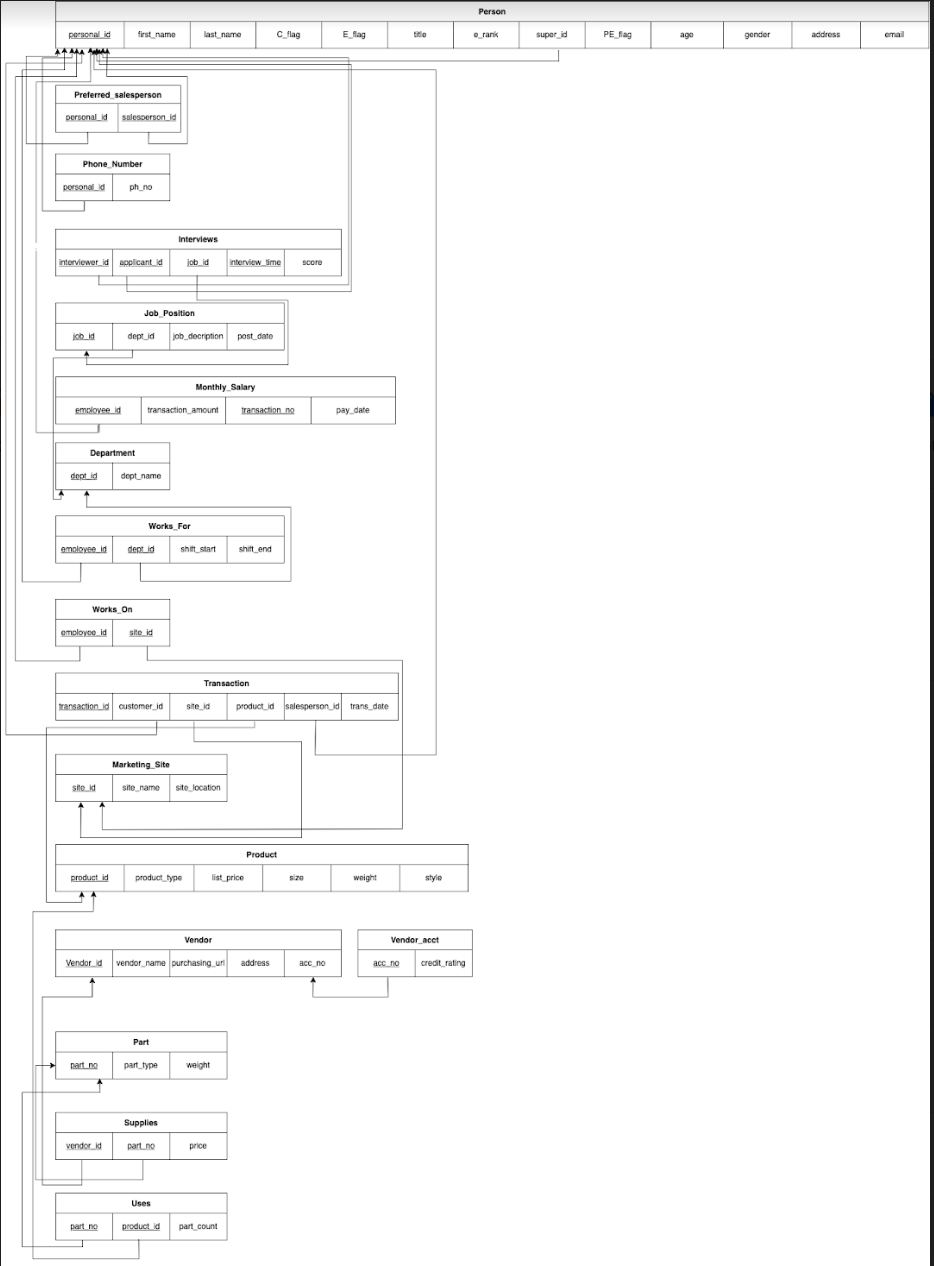


# Assumptions

1. In the company, the people can be divided into three subcategories, ‘Customer’, ‘Employee’, and ‘Potential Employee’, where a person must be in *at least one* of the subcategories. Salesperson is considered an employee of the company.
2. Address is a single string attribute that will always be written in a (Street name, City, State, Country) format.
3. When a customer makes an order they can only purchase one product. Transaction\_id was added as a surrogate key.
4. The Person relation will be used in 2nd normal form rather than 3rd for database functionality.
5. For View 4 we are assuming that the view should return the sum of the purchase cost of all the parts for a given product\_id.
6. Part number will be unique for each part type sold by different vendors.
7. Product price, size, weight, and style depend only on the product number, so these attributes can vary within the same product style.

# DB Schema

We used the EER diagram that we created and mapped it to a database schema, giving us the schema below. It is important to note that this is our finalized schema diagram after our design was normalized. The main relations affected by normalization changes were the Vendor relation and the Person relation. After normalizing all of them to 3NF, we chose to represent Person relationship in a lower form, leaving it in 2NF for the functionality purposes in MySQL implementation.



# Normalization Process

Now that we have developed our EER diagram and schema, we will go through the process of standardizing the schema into the third normal form.

## First Normal Form

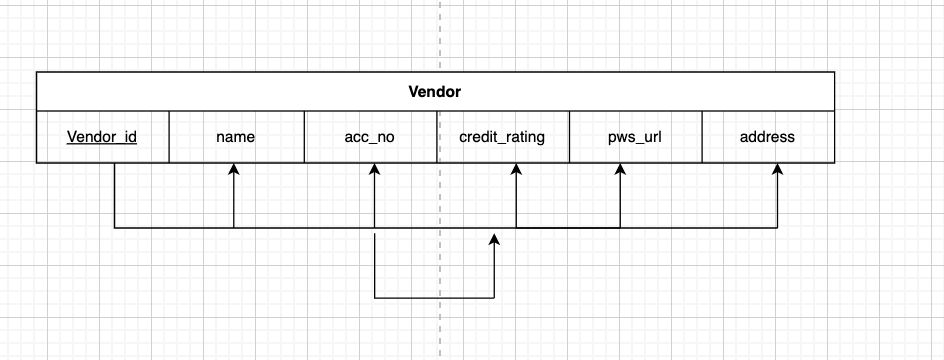
First normal form disallows the use of composite attributes, multivalued attributes, and nested relations. Our relation schema does not have any of these, so our relation schema is already in first normal form.

## Second Normal Form

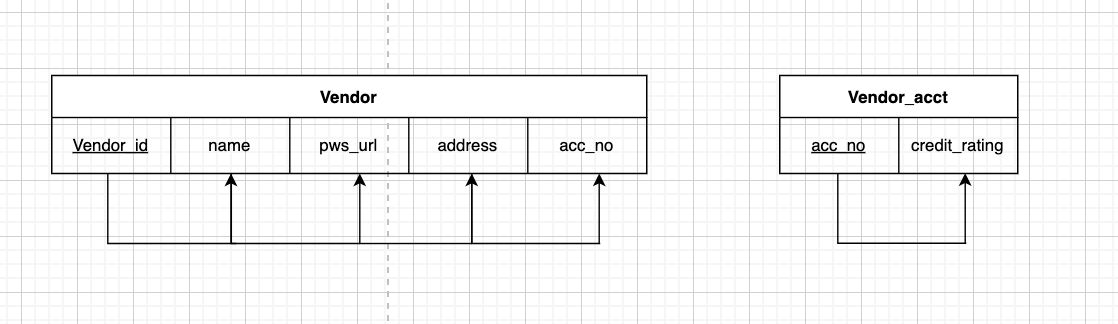
Second normal form is when every non-prime attribute A in R is fully dependent on the primary key. We know that if a PK is a single attribute, we do not need to apply a test and the relation is in second normal form. Looking through our dependency diagrams we see that we are operating within the paradigms of second normal form. Therefore, our relation schema is in second normal form without us having to make any changes.

## Third Normal Form

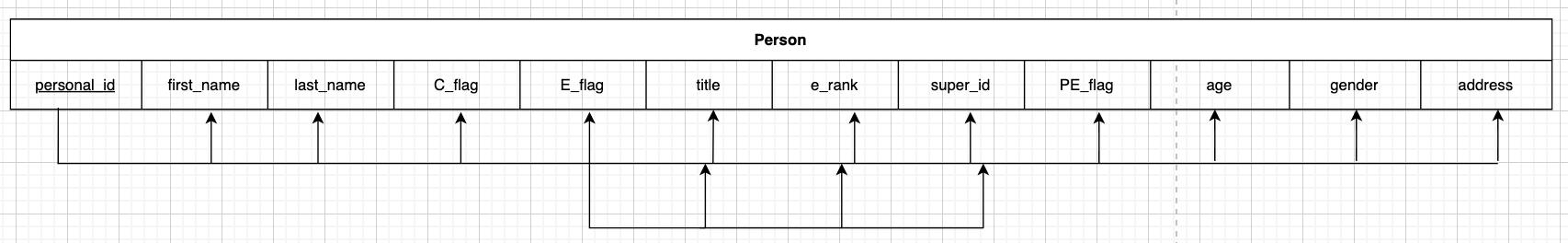
A relation schema is said to be in third normal form if it is in 2NF and no non-prime attribute A in R is transitively dependent on the primary key. We did have a couple violations of the third normal form in our relational schema for the vendor relation that can be seen below:



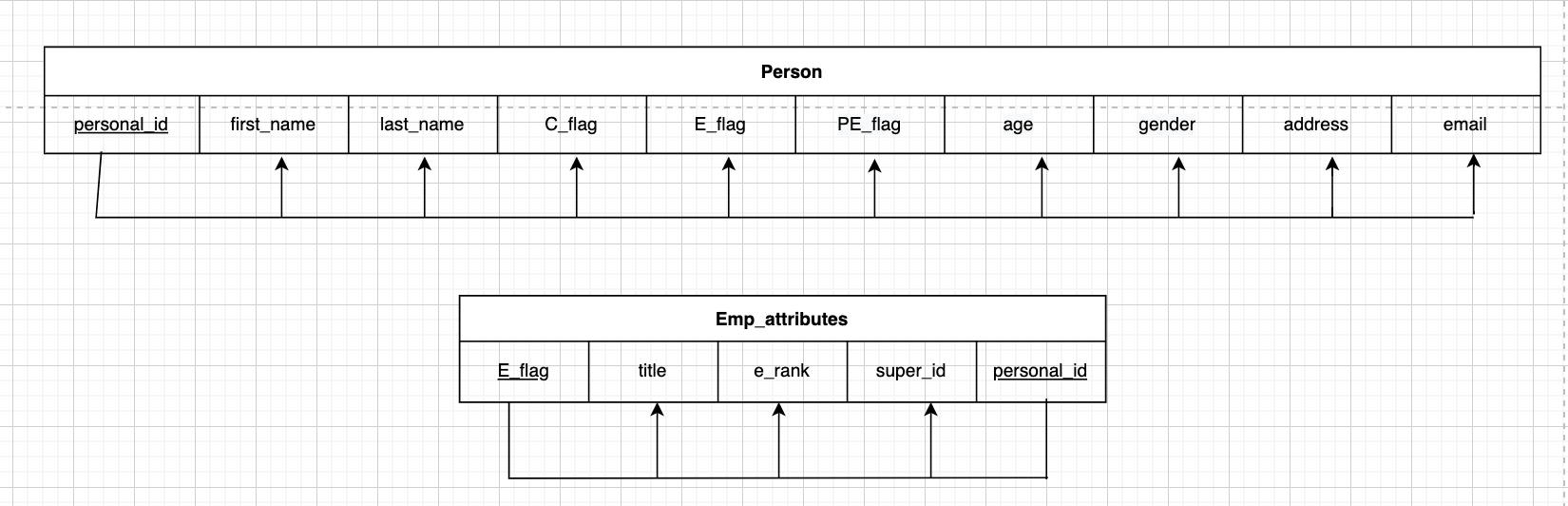
We can see here that the attribute credit\_rating is dependent on acc\_no, which indicates credit\_rating is dependent on acc\_no (stated in our assumptions). To fix this we created a Vendor\_acct table and separated the attributes that were causing violation of 3NF as seen below:



We also see that our Person relation is not in 3NF as title, e\_rank, and super\_id have a dependence on E\_flag. This is a violation of 3NF.



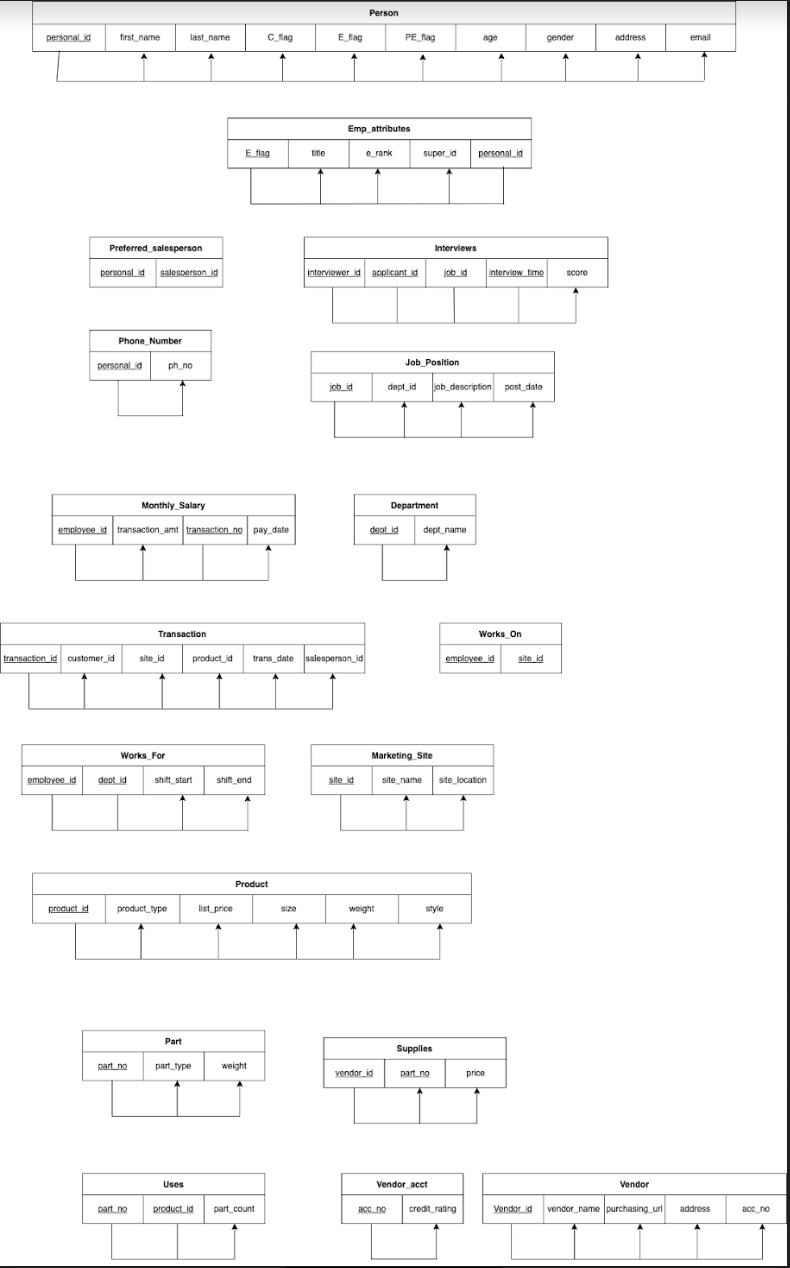
To fix the violation mentioned above we make the changes that can be seen below:



After making these changes our relational schema is now in 3NF. It is important to note that in our DB schema, we are using our person relation in 2NF rather than 3NF for better database functionality.

# Dependency Diagrams

Below we have the dependency diagrams created for all relations of our DB schema, which were used throughout the normalization process:



# MySQL Implementation

## Create Database

drop schema if exists company;

create database company;

use company;

create table person(

personal\_id int(10),

first\_name varchar(255),

last\_name varchar(255),

c\_flag boolean,

e\_flag boolean,

title varchar(255),

e\_rank varchar(255),

super\_id int(10),

pe\_flag boolean,

age int check (age < 65 or age is null),

gender varchar(255),

address varchar(255),

email varchar(255),

primary key (personal\_id),

foreign key (super\_id) references person(personal\_id),

check ((e\_flag is false and title is null and e\_rank is null and super\_id is null)

OR (e\_flag is true and title is not null and e\_rank is not null))

);

create table preferred\_salesperson(

personal\_id int(10),

salesperson\_id int(10),

foreign key (personal\_id) references person(personal\_id),

foreign key (salesperson\_id) references person(personal\_id)

);

create table phone\_no(

personal\_id int(10),

phone\_no int(10),

foreign key (personal\_id) references person(personal\_id)

);

create table monthly\_salary(

employee\_id int(10),

transaction\_amount double,

transaction\_no int,

pay\_date date,

primary key (employee\_id,transaction\_no),

foreign key (employee\_id) references person(personal\_id)

);

create table department(

dept\_id int,

dept\_name varchar(255),

primary key (dept\_id)

);

create table job\_position(

job\_id int,

dept\_id int,

job\_description varchar(225),

post\_date date,

primary key (job\_id),

foreign key (dept\_id) references department(dept\_id)

);

create table interviews(

interviewer\_id int(10),

applicant\_id int(10),

job\_id int,

interview\_time datetime,

score double check (score between 0 and 100),

foreign key (interviewer\_id) references person(personal\_id),

foreign key (applicant\_id) references person(personal\_id),

foreign key (job\_id) references job\_position(job\_id)

);

create table works\_for(

employee\_id int(10),

dept\_id int,

shift\_start time,

shift\_end time,

foreign key (employee\_id) references person(personal\_id),

foreign key (dept\_id) references job\_position(dept\_id),

check (shift\_start < shift\_end)

);

create table marketing\_site(

site\_id int,

site\_name varchar(255),

site\_location varchar(255),

primary key (site\_id)

);

create table works\_on(

employee\_id int(10),

site\_id int,

foreign key (employee\_id) references person(personal\_id),

foreign key (site\_id) references marketing\_site(site\_id)

);

create table product(

product\_id int,

product\_type varchar(255),

list\_price double,

size double,

weight double,

style varchar(255),

primary key (product\_id)

);

create table transaction(

transaction\_id int,

product\_id int,

customer\_id int(10),

site\_id int,

salesperson\_id int(10),

trans\_date datetime,

foreign key (customer\_id) references person(personal\_id),

foreign key (product\_id) references product(product\_id),

foreign key (site\_id) references marketing\_site(site\_id),

foreign key (salesperson\_id) references person(personal\_id),

primary key (transaction\_id)

);

create table vendor(

vendor\_id int,

vendor\_name varchar(255),

acc\_no int unique,

purchasing\_url varchar(255),

address varchar(255),

primary key (vendor\_id)

);

create table vendor\_account(

acc\_no int,

credit\_rating double,

foreign key (acc\_no) references vendor(acc\_no)

);

create table part(

part\_no int,

part\_type varchar(255),

weight double,

primary key (part\_no)

);

create table supplies(

vendor\_id int,

part\_no int,

price double,

foreign key (vendor\_id) references vendor(vendor\_id),

foreign key (part\_no) references part(part\_no)

);

create table uses(

part\_no int,

product\_id int,

part\_count int,

foreign key (part\_no) references part(part\_no),

foreign key (product\_id) references product(product\_id)

);

## Creating Views

create view view1 as

select employee\_id, avg(transaction\_amount) as avg\_salary

from monthly\_salary

group by employee\_id;

create view view2 as

select applicant\_id, job\_id, count(score > 60) as pass\_count

from interviews

group by applicant\_id, job\_id;

create view view3 as

select p.product\_type, count(t.product\_id)

from transaction t

join product p on p.product\_id = t.product\_id

group by p.product\_type;

create view view4 as

select u.product\_id, sum(u.part\_count\*s.price) as cost

from uses u

join supplies s on u.part\_no = s.part\_no;

## Answering Queries

-- 1. Return the ID and Name of interviewers who participate in interviews where the interviewee's name is "Hellen Cole" arranged for job "11111".

select i.interviewer\_id, p.last\_name

from interviews i

join person p on i.interviewer\_id = p.personal\_id

where job\_id = 11111

and applicant\_id = (

select personal\_id

from person

where first\_name = 'Hellen'

and last\_name = 'Cole');

-- 2. Return the ID of all jobs which are posted by department "Marketing" in January 2011.

select job\_id

from job\_position

where dept\_id = (

select dept\_id

from department

where dept\_name = 'Marketing')

and month(post\_date) = 1 and year(post\_date) = 2011;

-- 3. Return the ID and Name of the employees having no supervisees

select personal\_id, last\_name

from person

where e\_flag = TRUE

and personal\_id not in (

select distinct super\_id

from person);

-- 4. Return the Id and Location of the marketing sites with no sale records during March 2011.

select site\_id, site\_location

from marketing\_site

where site\_id not in (

select site\_id

from transaction

where month(trans\_date) = 3 and year(trans\_date) = 2011);

-- 5. Return the job's id and description, which does not hire a suitable person one month after it is posted.

select job\_id, job\_description

from job\_position

where job\_id not in (

select j.job\_id

from (

select \* from interviews

natural join job\_position

having max(interview\_time) between post\_date and date\_add(post\_date, interval 1 month)) j

group by j.job\_id

having count(j.interview\_time) >= 5

and avg(j.score) > 70);

-- 6. Return the ID and Name of the salespeople who have sold all product types whose price is above $200.

select personal\_id, last\_name

from person

where personal\_id in (

select salesperson\_id

from transaction

where (

select distinct product\_type

from product

where list\_price > 200)

in (

select distinct p.product\_type

from product p join transaction t on p.product\_id = t.product\_id

));

-- 7. Return the department's id and name, which has no job post during 1/1/2011 and 2/1/2011.

-- (I'm assuming this is asking about posts on 1/1 or 2/1, as opposed to between 1/1 and 2/1.)

select dept\_id, dept\_name

from department

where dept\_id not in (

select dept\_id

from job\_position

where post\_date = '2011-01-01' or post\_date = '2011-02-01');

-- 8. Return the ID, Name, and Department ID of the existing employees who apply for job "12345".

select w.employee\_id, p.last\_name, group\_concat(w.dept\_id) as dept\_id

from works\_for w

join person p on w.employee\_id = p.personal\_id

where w.employee\_id in (

select applicant\_id

from interviews

where job\_id = '12345');

-- 9. Return the best seller's type in the company (sold the most items).

select p.product\_type

from transaction t

join product p on p.product\_id = t.product\_id

group by p.product\_type

order by count(t.transaction\_id) desc limit 1;

-- 10. Return the product type whose net profit is highest in the company (money earned minus the part cost).

select p.product\_type

from uses u

inner join supplies s on u.part\_no = s.part\_no

inner join product p on p.product\_id = u.product\_id

inner join transaction t on t.product\_id = p.product\_id

group by p.product\_id

order by (p.list\_price - sum(u.part\_count\*s.price)) \* count(t.transaction\_id) desc limit 1;

-- 11. Return the name and id of the employees who have worked in all departments after being hired by the company.

select last\_name, personal\_id

from person

where personal\_id in (

select employee\_id

from works\_for

group by employee\_id

having count(distinct dept\_id) = (

select count(distinct dept\_id)

from works\_for));

-- 12. Return the name and email address of the interviewee who is selected.

select last\_name, email

from person

where personal\_id in (

select distinct applicant\_id

from interviews

group by applicant\_id, job\_id

having count(interview\_time) >= 5

and avg(score) > 70);

-- 13. Retrieve the names, phone numbers, and email addresses of the interviewees selected for all the jobs they apply for.

select p.last\_name, group\_concat(phone\_no.phone\_no) as phone\_no, p.email

from person p

inner join phone\_no on p.personal\_id = phone\_no.personal\_id

where pe\_flag = TRUE

and p.personal\_id not in (

select applicant\_id

from interviews

group by applicant\_id, job\_id

having count(interview\_time) < 5

or avg(score) > 70);

-- 14. Return the employee's name and id whose average monthly salary is the highest in the company.

select m.employee\_id, p.last\_name

from monthly\_salary m

join person p on m.employee\_id = p.personal\_id

group by m.employee\_id

order by avg(transaction\_amount) desc limit 1;

-- 15. Return the ID and Name of the vendor who supplies part whose name is "Cup" and weight is smaller than 4 pounds, and the price is lowest among all vendors.

select v.vendor\_id, v.vendor\_name

from vendor v

inner join supplies s on v.vendor\_id = s.vendor\_id

inner join part p on p.part\_no = s.part\_no

where p.part\_type = 'Cup'

and p.weight < 4

having min(s.price);