CSC3170 Assignment 3 with Solutions

> - This is an individual assignment and should be submitted by 5 p.m., 20 April 2022 via Blackboard.
> - You should submit a `solution.zip` file containing only the MySQL script files for the corresponding questions, namely `Q1.sql`, `Q2.sql`, ..., `Q16.sql`.
> - Any submission with an incorrect format might lead to 0 points as the grading script relies strictly on the correct format.
> - We set 25 points for Question 1, and 5 points each for Question 2 to Question 16.

1. Consider a database schema that stores the following information:
    - `employees`: This is a table that contains the basic information of any employee:
        - `EMPLOYEE_ID`: The employee's identifier that determines other attributes. A 6-digit integer.
        - `FIRST_NAME`: The first name of an employee (might be empty with only one word for the name information, where the name will be filled in the last name attribute). At most 20 characters.
        - `LAST_NAME`: The last name of an employee. It must not be empty. At most 25 characters.
        - `EMAIL`: The email of an employee. It must not be empty as a work email will be assigned (and the suffix can be omitted).
        - `PHONE_NUMBER`: The phone number of an employee. Might be empty. At most 20 characters, and character '.' is allowed for a phone number.
        - `HIRE_DATE`: The date when the employee was hired for the first time.
        - `JOB_ID`: The current job id of an employee. It should be of the same format as that of `jobs.JOB_ID`. One must have a current job information. A 10-character string.
        - `SALARY`: The real annual salary of the employee. Normally should be guided by the minimum and maximum salary information described in the current job description, while some special cases that exceed the limit are allowed. A float number with at most 8 digits to the left of the decimal point, and 2 to the right of it (If not pointed out specifically, the "salary" refers to "annual salary").
        - `COMMISSION_PCT`: The commission percentage. A float number from 0 to 100 with 2 digits to the right of the decimal point. It's the attribute usually not used, and the typical value should be 0.00 here.
        - `MANAGER_ID`: Every employee should have a manager that guide him or her in the real work. However, an manager might not have any staff. The attribute should have the same format as the `EMPLOYEE_ID`.

        - `DEPARTMENT_ID` : Describes the department that the employee belongs to. Any employee should register his or her department, while some virtual department might not have any employee registered.
    - `jobs` : This table describes the jobs an employee has currently or in the past.
        - `JOB_ID` : The job identifier that determines other attributes of a job. Use a string no more than 10 characters.
        - `JOB_TITLE` : The string that describes the title of a job. No more than 35 characters and every job should have a title.
        - `MIN_SALARY` : The minimum salary that a specific job could have. Not in high precision and described with a 6-digit integer. It might be empty.
        - `MAX_SALARY` : The maximum salary that a specific job could have. Not in high precision and described with a 6-digit integer. It might be empty.
    - `departments` : The departments of the company.
        - `DEPARTMENT_ID` : The department identifier that determine the other attributes.
        - `DEPARTMENT_NAME` : An attribute that should not be empty. The name of the department.
        - `MANAGER_ID` : Decribes the manager of this department. Note that it could be different from the manager of an employee, i.e. the manager of some employee might be or might not be the manager of some department. The format of this attribute should be the same as that of `employees.EMPLOYEE_ID` . A department might not have its manager. If a department has its manager, it should have exactly one manager.
        - `LOCATION_ID` : Help to find the location of the department. Every department should have a location. The format of this attribute is the same as that of `locations.LOCATION_ID` .
    - `locations` : This table describes the location of a department.
        - `LOCATION_ID` : The location identifier that determines other attributes. An integer with 4 digits.
        - `STREET_ADDRESS` : A string that is no more than 40 characters for street address. Might be empty.
        - `POSTAL_CODE` : A string that is no more than 12 characters for postal code. Might be empty.
        - `CITY` : Can not be empty. A string within 30 characters that describe the city of the location information.
        - `STATE_PROVINCE` : Can be empty. A string within 25 characters that describe the state or province of the location information.
        - `COUNTRY_ID` : An attribute having the same format as `countries.COUNTRY_ID` that can not be empty.
    - `countries` : The country that some address (location) locates in.
        - `COUNTRY_ID` : A string with only and exactly two characters that identifies some country.
        - `COUNTRY_NAME` : Should not be empty. The name of some country.

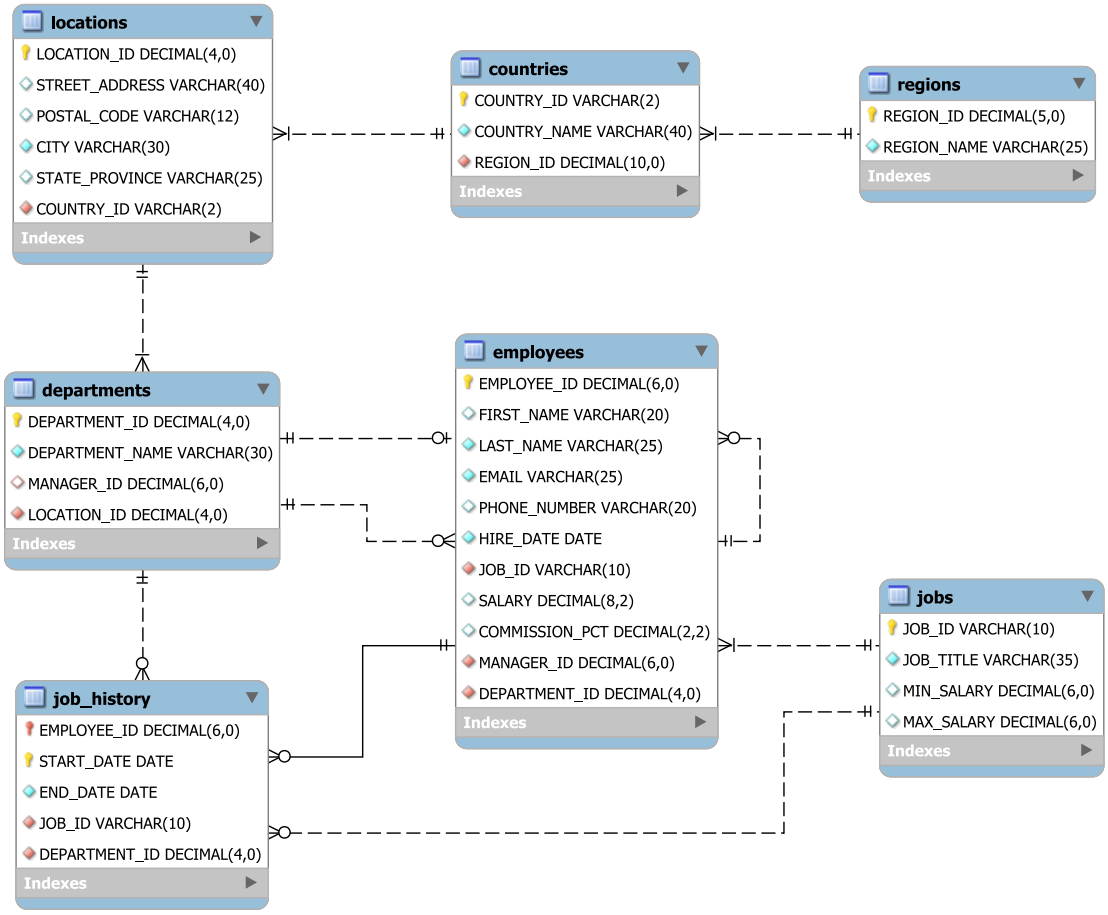
        - `REGION_ID` : Should not be empty. Have the same format as `regions.REGION_ID` .
    - `regions` : The region of some country.
        - `REGION_ID` : A 5-digit integer that identifies a region.
        - `REGION_NAME` : A string no more than 25 characters, which can not be empty.
    - `job_history` : An employee might have multiple job history records, while every job history should belong to only one employee.
        - `EMPLOYEE_ID` : The attribute having the same format as `employees.EMPLOYEE_ID` .
        - `START_DATE` : The start date of a job history record. Determines other attributes together with `EMPLOYEE_ID` in this table.
        - `END_DATE` : The end date of a job history record. Should not be empty.
        - `JOB_ID` : Describe the job type of a job history record. Have the same format as `jobs.JOB_ID` .
        - `DEPARTMENT_ID` : Help to describe the department information of a job history record. Have the same format as `departments.DEPARTMENT_ID` .

Please carefully check the information above, and write the MySQL scripts inside Q1.sql following the instruction in the comments. Note that this is one of the files you need to pack inside the `solution.zip` .

For the numerical data type like integer and float, you might just use `DECIMAL(M,D)` and adjust `M` and `D` , where `M` is the maximum number of digits (the precision) and `D` is the number of digits to the right of the decimal point. For the string, you might use `VARCHAR(N)` , where `N` is the maximum length. For the engine of the tables, you can use any suitable one provided by MySQL, but you need to consider the features like the foreign key support.

After finishing Q1.sql, you should execute it to see whether there is any error. When everything is okay, you need to execute db_insert.sql to check whether you can insert the test data. It consists of a feasible instance and you can use it to check whether your implementation from question 2 to 16 is correct with the test cases in the ./test_cases/ directory. If you can't execute db_insert.sql without errors, you might receive 0 points. The attributes in question 1 should be uppercase, while for question 2 to 16, you should also use uppercase (including the aggregation function names like `MAX` , `AVG` , etc.) though the words appeared in the questions are lowercase, for keeping consistent with the test cases.

Solution: The E-R diagram is shown as the following image:


locations
LOCATION_ID DECIMAL(4,0)
STREET_ADDRESS VARCHAR(40)
POSTAL_CODE VARCHAR(12)
CITY VARCHAR(30)
STATE_PROVINCE VARCHAR(25)
COUNTRY_ID VARCHAR(2)
Indexes
countries
COUNTRY_ID VARCHAR(2)
COUNTRY_NAME VARCHAR(40)
REGION_ID DECIMAL(10,0)
Indexes
regions
REGION_ID DECIMAL(5,0)
REGION_NAME VARCHAR(25)
Indexes
departments
DEPARTMENT_ID DECIMAL(4,0)
DEPARTMENT_NAME VARCHAR(30)
MANAGER_ID DECIMAL(6,0)
LOCATION_ID DECIMAL(4,0)
Indexes
employees
EMPLOYEE_ID DECIMAL(6,0)
FIRST_NAME VARCHAR(20)
LAST_NAME VARCHAR(25)
EMAIL VARCHAR(25)
PHONE_NUMBER VARCHAR(20)
HIRE_DATE DATE
JOB_ID VARCHAR(10)
SALARY DECIMAL(8,2)
COMMISSION_PCT DECIMAL(2,2)
MANAGER_ID DECIMAL(6,0)
DEPARTMENT_ID DECIMAL(4,0)
Indexes
jobs
JOB_ID VARCHAR(10)
JOB_TITLE VARCHAR(35)
MIN_SALARY DECIMAL(6,0)
MAX_SALARY DECIMAL(6,0)
Indexes
job_history
EMPLOYEE_ID DECIMAL(6,0)
START_DATE DATE
END_DATE DATE
JOB_ID VARCHAR(10)
DEPARTMENT_ID DECIMAL(4,0)
Indexes

You can check the table creation scripts in Q1_sol.sql, which is automatically generated by Workbench using the relationship described in the image shown above. Note that the foreign key constraints are not checked by the grading program, as `FOREIGN_KEY_CHECKS=0`. The indexing operations for speeding up future queries are also not required. To directly check the codes, you can see the fragment inserted below.

Also note that the uppercase and lowercase issues are not checked by the grading program, but it's still suggested to keep the case-consistency. The example codes for solutions to Q2-Q16 are directly attached right below the description of every question.

```
-- MySQL Script generated by MySQL Workbench
-- Mon Apr 25 15:52:46 2022
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema as3
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `as3` ;

-- -----------------------------------------------------
-- Schema as3
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `as3` DEFAULT CHARACTER SET utf8 ;
USE `as3` ;

-- -----------------------------------------------------
-- Table `as3`.`regions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`regions` (
  `REGION_ID` DECIMAL(5,0) NOT NULL,
  `REGION_NAME` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`REGION_ID`),
  UNIQUE INDEX `REG_NAME_IX` (`REGION_NAME` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `as3`.`countries`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`countries` (
  `COUNTRY_ID` VARCHAR(2) NOT NULL,
  `COUNTRY_NAME` VARCHAR(40) NOT NULL,
  `REGION_ID` DECIMAL(10,0) NOT NULL,
  PRIMARY KEY (`COUNTRY_ID`),
  INDEX `COUNTR_REG_FK` (`REGION_ID` ASC) VISIBLE,
  CONSTRAINT `fk_countries_regions`
    FOREIGN KEY (`REGION_ID`)
    REFERENCES `as3`.`regions` (`REGION_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `as3`.`locations`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`locations` (
  `LOCATION_ID` DECIMAL(4,0) NOT NULL DEFAULT '0',
  `STREET_ADDRESS` VARCHAR(40) NULL DEFAULT NULL,
  `POSTAL_CODE` VARCHAR(12) NULL DEFAULT NULL,
  `CITY` VARCHAR(30) NOT NULL,
  `STATE_PROVINCE` VARCHAR(25) NULL DEFAULT NULL,
  `COUNTRY_ID` VARCHAR(2) NOT NULL,
  PRIMARY KEY (`LOCATION_ID`),
  INDEX `LOC_CITY_IX` (`CITY` ASC) VISIBLE,
  INDEX `LOC_COUNTRY_IX` (`COUNTRY_ID` ASC) VISIBLE,
  INDEX `LOC_STATE_PROVINCE_IX` (`STATE_PROVINCE` ASC) VISIBLE,
  CONSTRAINT `FK_LOCATIONS_COUNTRIES`
    FOREIGN KEY (`COUNTRY_ID`)
    REFERENCES `as3`.`countries` (`COUNTRY_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `as3`.`jobs`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`jobs` (
  `JOB_ID` VARCHAR(10) NOT NULL DEFAULT '',
  `JOB_TITLE` VARCHAR(35) NOT NULL,
  `MIN_SALARY` DECIMAL(6,0) NULL DEFAULT NULL,
  `MAX_SALARY` DECIMAL(6,0) NULL DEFAULT NULL,
  PRIMARY KEY (`JOB_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `as3`.`employees`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`employees` (
  `EMPLOYEE_ID` DECIMAL(6,0) NOT NULL DEFAULT '0',
  `FIRST_NAME` VARCHAR(20) NULL DEFAULT NULL,
  `LAST_NAME` VARCHAR(25) NOT NULL,
  `EMAIL` VARCHAR(25) NOT NULL,
  `PHONE_NUMBER` VARCHAR(20) NULL DEFAULT NULL,
  `HIRE_DATE` DATE NOT NULL,
  `JOB_ID` VARCHAR(10) NOT NULL,
  `SALARY` DECIMAL(8,2) NULL DEFAULT NULL,
  `COMMISSION_PCT` DECIMAL(2,2) NULL DEFAULT NULL,
```

```
  `MANAGER_ID` DECIMAL(6,0) NOT NULL,
  `DEPARTMENT_ID` DECIMAL(4,0) NOT NULL,
  PRIMARY KEY (`EMPLOYEE_ID`),
  UNIQUE INDEX `EMP_EMAIL_UK` (`EMAIL` ASC) VISIBLE,
  INDEX `EMP_DEPARTMENT_IX` (`DEPARTMENT_ID` ASC) VISIBLE,
  INDEX `EMP_JOB_IX` (`JOB_ID` ASC) VISIBLE,
  INDEX `EMP_MANAGER_IX` (`MANAGER_ID` ASC) VISIBLE,
  INDEX `EMP_NAME_IX` (`LAST_NAME` ASC, `FIRST_NAME` ASC) VISIBLE,
  CONSTRAINT `FK_EMPLOYEES_JOBS`
    FOREIGN KEY (`JOB_ID`)
    REFERENCES `as3`.`jobs` (`JOB_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_EMPLOYEES_DEPARTMENTS`
    FOREIGN KEY (`DEPARTMENT_ID`)
    REFERENCES `as3`.`departments` (`DEPARTMENT_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `SK_MANAGER`
    FOREIGN KEY (`MANAGER_ID`)
    REFERENCES `as3`.`employees` (`EMPLOYEE_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `as3`.`departments`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`departments` (
  `DEPARTMENT_ID` DECIMAL(4,0) NOT NULL DEFAULT '0',
  `DEPARTMENT_NAME` VARCHAR(30) NOT NULL,
  `MANAGER_ID` DECIMAL(6,0) NULL DEFAULT NULL,
  `LOCATION_ID` DECIMAL(4,0) NOT NULL,
  PRIMARY KEY (`DEPARTMENT_ID`),
  INDEX `DEPT_MGR_FK` (`MANAGER_ID` ASC) VISIBLE,
  INDEX `DEPT_LOCATION_IX` (`LOCATION_ID` ASC) VISIBLE,
  CONSTRAINT `FK_DEPARTMENTS_LOCATIONS`
    FOREIGN KEY (`LOCATION_ID`)
    REFERENCES `as3`.`locations` (`LOCATION_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_DEPARTMENTS_EMPLOYEES_MANAGER`
    FOREIGN KEY (`MANAGER_ID`)
    REFERENCES `as3`.`employees` (`EMPLOYEE_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `as3`.`job_history`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `as3`.`job_history` (
  `EMPLOYEE_ID` DECIMAL(6,0) NOT NULL,
  `START_DATE` DATE NOT NULL,
  `END_DATE` DATE NOT NULL,
  `JOB_ID` VARCHAR(10) NOT NULL,
  `DEPARTMENT_ID` DECIMAL(4,0) NOT NULL,
  PRIMARY KEY (`EMPLOYEE_ID`, `START_DATE`),
  INDEX `JHIST_DEPARTMENT_IX` (`DEPARTMENT_ID` ASC) VISIBLE,
  INDEX `JHIST_EMPLOYEE_IX` (`EMPLOYEE_ID` ASC) VISIBLE,
  INDEX `JHIST_JOB_IX` (`JOB_ID` ASC) VISIBLE,
  CONSTRAINT `FK_JOB_HISTORY_DEPARTMENTS`
    FOREIGN KEY (`DEPARTMENT_ID`)
    REFERENCES `as3`.`departments` (`DEPARTMENT_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_JOB_HISTORY_EMPLOYEES`
    FOREIGN KEY (`EMPLOYEE_ID`)
    REFERENCES `as3`.`employees` (`EMPLOYEE_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FK_JOB_HISTORY_JOBS`
    FOREIGN KEY (`JOB_ID`)
    REFERENCES `as3`.`jobs` (`JOB_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

2. Show the first_name and last_name of all employees using alias name "First Name", "Last Name".

```
SELECT FIRST_NAME "First Name", LAST_NAME "Last Name"
FROM employees;
```

3. Write a query to show the employee_id and the salary of all employees in ascending order of salary.

```
SELECT EMPLOYEE_ID, SALARY
FROM employees
ORDER BY salary;
```

4. Write a query to calculate the maximum and minimum salary of all employees.

```
SELECT MAX(SALARY), MIN(SALARY)
FROM employees;
```

5. Show the employee_id and the monthly salary (round 2 decimal places) of all employees.

```
SELECT EMPLOYEE_ID, ROUND(SALARY/12, 2)
FROM employees;
```

6. Show the employee(s) that is the manager of some department, but doesn't manages any other employees.

```
SELECT EMPLOYEE_ID from employees
WHERE
EMPLOYEE_ID in (SELECT MANAGER_ID from departments)
and
EMPLOYEE_ID not in (select MANAGER_ID from employees);
```

7. Write a query that selects employee_id, and phone_number of those employees who are in departments 20 or 100. The results should be descendingly ordered by the department_id.

```
SELECT EMPLOYEE_ID, PHONE_NUMBER
FROM employees
WHERE DEPARTMENT_ID IN (20, 100)
ORDER BY DEPARTMENT_ID desc;
```

8. Write a query that selects the first_name of employees having 'a' as the second character.

```
SELECT FIRST_NAME
FROM employees
WHERE FIRST_NAME LIKE '_a%';
```

9. Write a query that get the number of employees who have the same job. The results should contain job_id and its number of employees.

```
SELECT JOB_ID, COUNT(*)
FROM employees
GROUP BY JOB_ID;
```

10. Write a query to calculate the average salary for all departments that have over 10 employees. The results should contain department_id, the corresponding average salary, and the number of employees.

```
SELECT DEPARTMENT_ID, AVG(SALARY), COUNT(*)
FROM employees
GROUP BY DEPARTMENT_ID
HAVING COUNT(*) > 10;
```

11. Write a query to get the first_name and last_name of the employees who have a manager that is currently working in a USA based department.

```
SELECT FIRST_NAME, LAST_NAME FROM employees
WHERE MANAGER_ID in (select EMPLOYEE_ID
FROM employees WHERE DEPARTMENT_ID
IN (SELECT DEPARTMENT_ID FROM departments WHERE LOCATION_ID
IN (select LOCATION_ID from locations where COUNTRY_ID='US')));
```

12. Write a query to get the employee_id and salary of the employees whose salary is larger than the average salary of all departments.

```
SELECT EMPLOYEE_ID, SALARY FROM employees
WHERE SALARY >
ALL(SELECT avg(SALARY) FROM employees GROUP BY DEPARTMENT_ID);
```

13. Write a query to find the 4th lowest salary of employees. The results should also contain employee_id and salary.

```
SELECT DISTINCT EMPLOYEE_ID, SALARY
FROM employees e1
WHERE 4 = (SELECT COUNT(DISTINCT SALARY)
FROM employees e2
WHERE e2.SALARY <= e1.SALARY);
```

14. Write a query to find the employee_id, job, department's id and name of the employees working in Seattle.

```
SELECT e.EMPLOYEE_ID, e.JOB_ID, e.DEPARTMENT_ID, d.DEPARTMENT_NAME
FROM employees e
JOIN departments d
ON (e.DEPARTMENT_ID = d.DEPARTMENT_ID)
JOIN locations l ON
(d.LOCATION_ID = l.LOCATION_ID)
WHERE LOWER(l.CITY) = 'seattle';
```

15. Write a query to get the department's id and the number of employees in the department. The results should contain two keys: "Department Name" and "Number of Employees".

```sql
SELECT departments.DEPARTMENT_ID AS 'Department Name',
COUNT(*) AS 'Number of Employees'
FROM departments
INNER JOIN employees
ON employees.DEPARTMENT_ID = departments.DEPARTMENT_ID
GROUP BY departments.DEPARTMENT_ID, DEPARTMENT_NAME
ORDER BY DEPARTMENT_NAME;
```

16. Write a query to get the department_id, department_name, and manager's first_name for departments.

```sql
SELECT d.DEPARTMENT_ID, d.DEPARTMENT_NAME, e.FIRST_NAME
FROM departments d
INNER JOIN employees e
ON (d.MANAGER_ID = e.EMPLOYEE_ID);
```