Final Review Project

1. Let's review how we create tables. Create a table named student1 with 4 columns:
   - student_id that stores INTEGER
   - name that stores TEXT
   - major that stores Text
   - GPA that stores Decimal, 2 digits with 1 decimal point

```
mysql> CREATE TABLE student(student_id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> name TEXT NOT NULL,
    -> major TEXT,
    -> GPA Decimal(2,1));
Query OK, 0 rows affected (0.05 sec)

mysql> describe student;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| student_id | int(11)     | NO   | PRI | NULL    | auto_increment |
| name       | text        | NO   |     | NULL    |                |
| major      | text        | YES  |     | NULL    |                |
| GPA        | decimal(2,1) | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

   This time, you will need to use the **constraints**.

   1) Find how to set student_id is auto-generated and incremented, so you do not need to put the id number in your insert query every time when you insert data in student.

      i.e. INSERT INTO student (name, major, GPA) VALUES ('Robert', 'Biology');

```
mysql> INSERT INTO student(name, major, GPA)
    -> VALUES('Robert', 'Biology', 3.9);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM student;
+------------+--------+---------+------+
| student_id | name   | major   | GPA  |
+------------+--------+---------+------+
|          1 | Robert | Biology |  3.9 |
+------------+--------+---------+------+
1 row in set (0.00 sec)
```

   2) Also, make sure the name column would not null. Try to insert one student with name=NULL, and see the error. Copy the result in your submission.

```
mysql> INSERT INTO student
    -> VALUES(NULL, 'Math', 3.8);
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql>
```

3) Put four more students info in your table (Name from anyone in our class, Major = Computer Science, Math, History, Biology).

```
mysql> INSERT INTO student(name, major, GPA)
    -> VALUES('Kyle Stewart', 'Computer Science', 4.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student(name, major, GPA)
    -> VALUES('Chris Ang', 'Computer Science', 4.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student(name, major, GPA)
    -> VALUES('Marina Saldana', 'Biology', 3.7);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student(name, major, GPA)
    -> VALUES('Gianni Castellanos', 'Math', 4.0);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student(name, major, GPA)
    -> VALUES('Gianni ^Cstellanos', 'Math', 4.0);
mysql> select*from student;
+------------+--------------------+------------------+------+
| student_id | name               | major            | GPA  |
+------------+--------------------+------------------+------+
|          1 | Robert             | Biology          |  3.9 |
|          2 | Kyle Stewart       | Computer Science |  4.0 |
|          3 | Chris Ang          | Computer Science |  4.0 |
|          4 | Marina Saldana     | Biology          |  3.7 |
|          5 | Gianni Castellanos | Math             |  4.0 |
+------------+--------------------+------------------+------+
```

4) Set the primary key.
   Refer back to number 1
5) Use SELECT * to show your table.
   Refer back to number 1
6) Use DESCRIBE to show the structure of the table.
   Refer back to number 1

2. Starting from the 2019 year, we will name Biology as Bio-Tech. Change the name of the major name Biology to Bio-Tech.

```
mysql> UPDATE student
    -> SET major = 'bio-tech'
    -> WHERE major = 'biology';
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select*from student;
+------------+--------------------+------------------+------+
| student_id | name               | major            | GPA  |
+------------+--------------------+------------------+------+
|          1 | Robert             | bio-tech         |  3.9 |
|          2 | Kyle Stewart       | Computer Science |  4.0 |
|          3 | Chris Ang          | Computer Science |  4.0 |
|          4 | Marina Saldana     | bio-tech         |  3.7 |
|          5 | Gianni Castellanos | Math             |  4.0 |
+------------+--------------------+------------------+------+
5 rows in set (0.00 sec)
```

UPDATED:

```
mysql> UPDATE student
    -> SET major = 'history'
    -> WHERE student_id = 4;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select*from student;
+------------+--------------------+------------------+------+
| student_id | name               | major            | GPA  |
+------------+--------------------+------------------+------+
|          1 | Robert             | bio-Tech         |  3.9 |
|          2 | Kyle Stewart       | Computer Science |  4.0 |
|          3 | Chris Ang          | Computer Science |  4.0 |
|          4 | Marina Saldana     | history          |  3.7 |
|          5 | Gianni Castellanos | Math             |  4.0 |
+------------+--------------------+------------------+------+
5 rows in set (0.00 sec)
```

3. Show the names of students whose major is either Math or Computer Science.

```
mysql> SELECT name FROM student
    -> WHERE major = 'Math' or major = 'Computer Science';
+--------------------+
| name               |
+--------------------+
| Kyle Stewart       |
| Chris Ang          |
| Gianni Castellanos |
+--------------------+
3 rows in set (0.00 sec)
```

4. Delete the second student information who major the Bio-Tech.

```
mysql> DELETE FROM student
    -> WHERE major = 'bio-Tech';
Query OK, 1 row affected (0.00 sec)

mysql> select * from student;
+------------+--------------------+------------------+------+
| student_id | name               | major            | GPA  |
+------------+--------------------+------------------+------+
|          2 | Kyle Stewart       | Computer Science |  4.0 |
|          3 | Chris Ang          | Computer Science |  4.0 |
|          4 | Marina Saldana     | history          |  3.7 |
|          5 | Gianni Castellanos | Math             |  4.0 |
+------------+--------------------+------------------+------+
4 rows in set (0.00 sec)
```

5. Add one more column named address with TEXT (WAS INTEGER PREVIOUSLY).

```
mysql> ALTER TABLE student
    -> ADD address INTEGER;
Query OK, 4 rows affected (0.07 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------------+--------------------+------------------+------+---------+
| student_id | name               | major            | GPA  | address |
+------------+--------------------+------------------+------+---------+
|          2 | Kyle Stewart       | Computer Science |  4.0 |    NULL |
|          3 | Chris Ang          | Computer Science |  4.0 |    NULL |
|          4 | Marina Saldana     | history          |  3.7 |    NULL |
|          5 | Gianni Castellanos | Math             |  4.0 |    NULL |
+------------+--------------------+------------------+------+---------+
4 rows in set (0.00 sec)
```

6. Delete the address column.

```
mysql> ALTER TABLE student
    -> DROP COLUMN address;
Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------------+--------------------+------------------+------+
| student_id | name               | major            | GPA  |
+------------+--------------------+------------------+------+
|          2 | Kyle Stewart       | Computer Science |  4.0 |
|          3 | Chris Ang          | Computer Science |  4.0 |
|          4 | Marina Saldana     | history          |  3.7 |
|          5 | Gianni Castellanos | Math             |  4.0 |
+------------+--------------------+------------------+------+
4 rows in set (0.00 sec)
```

7. Sort the name in alphabetical order (A-Z).

```
mysql> SELECT * FROM student
    -> ORDER BY name;
+------------+--------------------+------------------+------+
| student_id | name               | major            | GPA  |
+------------+--------------------+------------------+------+
|          3 | Chris Ang          | Computer Science |  4.0 |
|          5 | Gianni Castellanos | Math             |  4.0 |
|          2 | Kyle Stewart       | Computer Science |  4.0 |
|          4 | Marina Saldana     | history          |  3.7 |
+------------+--------------------+------------------+------+
4 rows in set (0.00 sec)

mysql> SELECT name FROM student
    -> ORDER BY name;
+--------------------+
| name               |
+--------------------+
| Chris Ang          |
| Gianni Castellanos |
| Kyle Stewart       |
| Marina Saldana     |
+--------------------+
4 rows in set (0.00 sec)
```

8. Show the top 2 highest GPA students

```
mysql> SELECT * FROM student
    -> ORDER BY GPA DESC LIMIT 2;
+------------+--------------+------------------+------+
| student_id | name         | major            | GPA  |
+------------+--------------+------------------+------+
|          2 | Kyle Stewart | Computer Science |  4.0 |
|          3 | Chris Ang    | Computer Science |  4.0 |
+------------+--------------+------------------+------+
2 rows in set (0.00 sec)
```

9. Show the students who major Computer Science and the GPA is higher than 3.5

```
mysql> SELECT * FROM student
    -> WHERE major = 'Computer Science' AND GPA > 3.5;
+------------+--------------+------------------+------+
| student_id | name         | major            | GPA  |
+------------+--------------+------------------+------+
|          2 | Kyle Stewart | Computer Science |  4.0 |
|          3 | Chris Ang    | Computer Science |  4.0 |
+------------+--------------+------------------+------+
2 rows in set (0.00 sec)
```

10. Use the keyword IN to show the students name who major Math or History

```
mysql> SELECT * FROM student
    -> WHERE major IN ('Math','history');
+------------+--------------------+---------+------+
| student_id | name               | major   | GPA  |
+------------+--------------------+---------+------+
|          4 | Marina Saldana     | history |  3.7 |
|          5 | Gianni Castellanos | Math    |  3.8 |
+------------+--------------------+---------+------+
2 rows in set (0.03 sec)
```

From the Works_With, Employee, Branch, Client tables,

11. Find all employees sorted by sex, then name

```
mysql> select * from employee
    -> GROUP BY sex,first_name,last_name;
+--------+------------+-----------+------------+------+--------+----------+-----------+
| emp_id | first_name | last_name | birth_date | sex  | salary | super_id | branch_id |
+--------+------------+-----------+------------+------+--------+----------+-----------+
|    103 | Angela     | Martin    | 1971-06-25 | F    |  63000 | 102      |         2 |
|    101 | Jan        | Levinson  | 1961-05-11 | F    | 110000 | 100      |         1 |
|    104 | Kelly      | Kapoor    | 1980-02-05 | F    |  55000 | 102      |         2 |
|    107 | Andy       | Bernard   | 1973-07-22 | M    |  65000 | 106      |         3 |
|    100 | David      | Wallace   | 1967-11-17 | M    | 250000 |          |         1 |
|    108 | Jim        | Halpert   | 1978-10-01 | M    |  71000 | 106      |         3 |
|    106 | Josh       | Porter    | 1969-09-05 | M    |  78000 | 100      |         3 |
|    102 | Michael    | Scott     | 1964-03-15 | M    |  75000 | 100      |         2 |
|    105 | Stanley    | Hudson    | 1958-02-19 | M    |  69000 | 102      |         2 |
+--------+------------+-----------+------------+------+--------+----------+-----------+
9 rows in set (0.00 sec)
```

12. Find the first name and last name of all employee. Change the title of each column to forename and surnames.

```
mysql> select first_name, last_name FROM employee;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| David      | Wallace   |
| Jan        | Levinson  |
| Michael    | Scott     |
| Angela     | Martin    |
| Kelly      | Kapoor    |
| Stanley    | Hudson    |
| Josh       | Porter    |
| Andy       | Bernard   |
| Jim        | Halpert   |
+------------+-----------+
9 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE employee
    -> CHANGE first_name forename TEXT;
Query OK, 9 rows affected (0.06 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE employee
    -> CHANGE last_name surname TEXT;
Query OK, 9 rows affected (0.05 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select*from employee;
+--------+----------+----------+------------+------+--------+----------+-----------+
| emp_id | forename | surname  | birth_date | sex  | salary | super_id | branch_id |
+--------+----------+----------+------------+------+--------+----------+-----------+
|    100 | David    | Wallace  | 1967-11-17 | M    | 250000 |          |         1 |
|    101 | Jan      | Levinson | 1961-05-11 | F    | 110000 | 100      |         1 |
|    102 | Michael  | Scott    | 1964-03-15 | M    |  75000 | 100      |         2 |
|    103 | Angela   | Martin   | 1971-06-25 | F    |  63000 | 102      |         2 |
|    104 | Kelly    | Kapoor   | 1980-02-05 | F    |  55000 | 102      |         2 |
|    105 | Stanley  | Hudson   | 1958-02-19 | M    |  69000 | 102      |         2 |
|    106 | Josh     | Porter   | 1969-09-05 | M    |  78000 | 100      |         3 |
|    107 | Andy     | Bernard  | 1973-07-22 | M    |  65000 | 106      |         3 |
|    108 | Jim      | Halpert  | 1978-10-01 | M    |  71000 | 106      |         3 |
+--------+----------+----------+------------+------+--------+----------+-----------+
9 rows in set (0.00 sec)
```

13. Find all the different branch_id.

```
mysql> SELECT DISTINCT branch_id FROM employee;
+-----------+
| branch_id |
+-----------+
|         1 |
|         2 |
|         3 |
+-----------+
3 rows in set (0.00 sec)
```

14. The sup_id means the supervisor id. Now, find the number of employees who has supervisor.

```
mysql> UPDATE employee
    -> SET super_id = NULL
    -> WHERE super_id = " ";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select*from employee;
+--------+----------+----------+------------+------+--------+----------+-----------+
| emp_id | forename | surname  | birth_date | sex  | salary | super_id | branch_id |
+--------+----------+----------+------------+------+--------+----------+-----------+
|    100 | David    | Wallace  | 1967-11-17 | M    | 250000 | NULL     |         1 |
|    101 | Jan      | Levinson | 1961-05-11 | F    | 110000 | 100      |         1 |
|    102 | Michael  | Scott    | 1964-03-15 | M    |  75000 | 100      |         2 |
|    103 | Angela   | Martin   | 1971-06-25 | F    |  63000 | 102      |         2 |
|    104 | Kelly    | Kapoor   | 1980-02-05 | F    |  55000 | 102      |         2 |
|    105 | Stanley  | Hudson   | 1958-02-19 | M    |  69000 | 102      |         2 |
|    106 | Josh     | Porter   | 1969-09-05 | M    |  78000 | 100      |         3 |
|    107 | Andy     | Bernard  | 1973-07-22 | M    |  65000 | 106      |         3 |
|    108 | Jim      | Halpert  | 1978-10-01 | M    |  71000 | 106      |         3 |
+--------+----------+----------+------------+------+--------+----------+-----------+
9 rows in set (0.00 sec)

mysql> SELECT COUNT(super_id) FROM employee;
+-----------------+
| COUNT(super_id) |
+-----------------+
|               8 |
+-----------------+
1 row in set (0.00 sec)
```

15. Find the number of female employees born after 1970

```
mysql> SELECT COUNT(sex) FROM employee
    -> WHERE sex = 'F' AND birth_date > '1970';
+-------------+
| COUNT(sex) |
+-------------+
|           2 |
+-------------+
1 row in set (0.00 sec)
```

16. Find the first and last name of the employee whose salary is more than the average salary.

```
mysql> SELECT forename, surname FROM employee WHERE salary > (SELECT AVG(salary) FROM employee);
+----------+----------+
| forename | surname  |
+----------+----------+
| David    | Wallace  |
| Jan      | Levinson |
+----------+----------+
2 rows in set (0.00 sec)
```

17. Find the sum of all employee's salaries

```
mysql> SELECT SUM(salary) FROM employee;
+-------------+
| SUM(salary) |
+-------------+
|      836000 |
+-------------+
1 row in set (0.00 sec)
```

18. Find how many males and females in employee

```
mysql> SELECT sex, COUNT(*) FROM employee
    -> GROUP BY sex;
+------+----------+
| sex  | COUNT(*) |
+------+----------+
| F    |        3 |
| M    |        6 |
+------+----------+
2 rows in set (0.00 sec)
```

19. Find the total sales of each employee. i.e how much each employee sold?

```
mysql> SELECT employee.emp_id, forename, surname, SUM(total_sales)
    -> FROM employee
    -> INNER JOIN works_with
    -> ON employee.emp_id = works_with.emp_id
    -> GROUP BY employee.emp_id;
+--------+----------+---------+------------------+
| emp_id | forename | surname | SUM(total_sales) |
+--------+----------+---------+------------------+
|    102 | Michael  | Scott   |           282000 |
|    105 | Stanley  | Hudson  |           218000 |
|    107 | Andy     | Bernard |            31000 |
|    108 | Jim      | Halpert |            34500 |
+--------+----------+---------+------------------+
4 rows in set (0.03 sec)
```

20. Find the total amount each client spent. Who are the top 2 clients?

```
mysql> SELECT client_name, SUM(total_sales) AS totals
    -> FROM client
    -> INNER JOIN works_with
    -> ON client.client_id = works_with.client_id
    -> GROUP BY works_with.client_id
    -> ORDER BY SUM(total_sales) DESC LIMIT 2;
+--------------------------+--------+
| client_name              | totals |
+--------------------------+--------+
| Renton Technical College | 267000 |
| QFC                      | 145000 |
+--------------------------+--------+
2 rows in set (0.02 sec)
```

21. Find any employee born in March

1 WARNING: forcing to convert a date data type to a string

```
mysql> select * from employee WHERE birth_date LIKE '____-03-__';
+--------+----------+---------+------------+------+--------+----------+-----------+
| emp_id | forename | surname | birth_date | sex  | salary | super_id | branch_id |
+--------+----------+---------+------------+------+--------+----------+-----------+
|    102 | Michael  | Scott   | 1964-03-15 | M    |  75000 | 100      |         2 |
+--------+----------+---------+------------+------+--------+----------+-----------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> ALTER TABLE employee
    -> MODIFY COLUMN birth_date varchar(10);
Query OK, 9 rows affected (0.04 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select * from employee WHERE birth_date LIKE '____-03-__';
+--------+----------+---------+------------+------+--------+----------+-----------+
| emp_id | forename | surname | birth_date | sex  | salary | super_id | branch_id |
+--------+----------+---------+------------+------+--------+----------+-----------+
|    102 | Michael  | Scott   | 1964-03-15 | M    |  75000 | 100      |         2 |
+--------+----------+---------+------------+------+--------+----------+-----------+
1 row in set (0.00 sec)
```

22. Find a list of employee, branch, and client name and name it as Company Names

```
mysql> SELECT forename AS Company_Name from employee
    -> UNION SELECT
    -> branch_name FROM branch
    -> UNION SELECT
    -> client_name FROM client;
+--------------------------+
| Company_Name             |
+--------------------------+
| David                    |
| Jan                      |
| Michael                  |
| Angela                   |
| Kelly                    |
| Stanley                  |
| Josh                     |
| Andy                     |
| Jim                      |
| Corporate                |
| Scranton                 |
| Stamford                 |
| Renton High School       |
| Renton Technical College |
| Bellevue College         |
| Hazen High School        |
| New York Times           |
| Seattle Times            |
| QFC                      |
+--------------------------+
19 rows in set (0.00 sec)
```

23. Find a list of total money spent by client(name the column as Sales) AND total money Earned by the employee (name the column as Salary).

```
mysql> SELECT * FROM (SELECT SUM(total_sales) as Sales from works_with) as T1 JOIN (SELECT SUM(salary) as
 Salary from employee) as T2;
+--------+--------+
| Sales  | Salary |
+--------+--------+
| 565500 | 836000 |
+--------+--------+
1 row in set (0.00 sec)
```

24. Let's insert one more row in branch table.
    INSERT INTO branch VALUES (4, 'Buffalo', NULL, NULL);

```
mysql> select*from branch;
+-----------+-------------+--------+----------------+
| branch_id | branch_name | mgr_id | mgr_start_date |
+-----------+-------------+--------+----------------+
|         1 | Corporate   |    100 | 2006-02-09     |
|         2 | Scranton    |    102 | 1992-04-06     |
|         3 | Stamford    |    106 | 1998-02-13     |
|         4 | Buffalo     |   NULL | NULL           |
+-----------+-------------+--------+----------------+
4 rows in set (0.00 sec)
```

25. Find all branches and the name of their manager

```
mysql> SELECT branch.branch_name, employee.forename FROM branch
    -> INNER JOIN employee ON branch.mgr_id = employee.emp_id;
+-------------+----------+
| branch_name | forename |
+-------------+----------+
| Corporate   | David    |
| Scranton    | Michael  |
| Stamford    | Josh     |
+-------------+----------+
3 rows in set (0.00 sec)
```

26. Perform the left join of employee table and branch table. Show only emp_id, first_name, and branch_name

```
mysql> SELECT employee.emp_id, employee.forename, branch.branch_name
    -> FROM employee
    -> LEFT JOIN branch ON employee.branch_id = branch.branch_id;
+--------+----------+-------------+
| emp_id | forename | branch_name |
+--------+----------+-------------+
|    100 | David    | Corporate   |
|    101 | Jan      | Corporate   |
|    102 | Michael  | Scranton    |
|    103 | Angela   | Scranton    |
|    104 | Kelly    | Scranton    |
|    105 | Stanley  | Scranton    |
|    106 | Josh     | Stamford    |
|    107 | Andy     | Stamford    |
|    108 | Jim      | Stamford    |
+--------+----------+-------------+
9 rows in set (0.00 sec)
```

27. Perform the right join of employee table and branch table. Show only emp_id, first_name, and branch_name

```
mysql> SELECT employee.emp_id, employee.forename, branch.branch_name
    -> FROM employee
    -> RIGHT JOIN branch ON employee.branch_id = branch.branch_id;
+--------+----------+-------------+
| emp_id | forename | branch_name |
+--------+----------+-------------+
|    100 | David    | Corporate   |
|    101 | Jan      | Corporate   |
|    102 | Michael  | Scranton    |
|    103 | Angela   | Scranton    |
|    104 | Kelly    | Scranton    |
|    105 | Stanley  | Scranton    |
|    106 | Josh     | Stamford    |
|    107 | Andy     | Stamford    |
|    108 | Jim      | Stamford    |
|   NULL | NULL     | Buffalo     |
+--------+----------+-------------+
10 rows in set (0.00 sec)
```

28. Compare the results of 25 and 26. Notice the values of NULL

    LEFT JOIN – gets the value of the first table (no null)

    RIGHT JOIN – gets the value of the second table (there are null)

29. Make your own scenario to use **HAVING** keyword

    Find the male and female employees who make less than $70,000 as their salary

```
mysql> SELECT sex, count(*) FROM employee
    -> WHERE salary < 70000
    -> GROUP by sex
    -> HAVING COUNT(sex) < 3;
+------+----------+
| sex  | count(*) |
+------+----------+
| F    |        2 |
| M    |        2 |
+------+----------+
2 rows in set (0.00 sec)
```

30. Make your own scenario to use the nested queries

    Find the employees who had sales less than $115,000

```
mysql> select employee.emp_id from employee
    -> where employee.emp_id
    -> IN (SELECT works_with.emp_id FROM works_with WHERE works_with.total_sales < 115000);
+--------+
| emp_id |
+--------+
|    105 |
|    108 |
|    107 |
|    102 |
+--------+
4 rows in set (0.00 sec)
```