SQL 基础练习题

1)下面的 SELECT 语句是否执行成功：SELECT last_name, job_id, salary AS Sal FROM employees;

1) SELECT 语句应该执行成功。它选择了 EMPLOYEES 表中的 last_name, job_id 和 salary 列，并将 salary 列的别名设置为 Sal。

2) 在下面的语句中有 4 个编码错误，请找出它们：SELECT employee_id, last_namesalx12 ANNUAL SALARYFROM employees;

2) 下面是修正后的语句，已经修复了所有的编码错误：SELECT employee_id, last_name, 'ANNUAL SALARY' AS salx12FROM employees;

3)显示 DEPARTMENTS 表的结构。选择表中的所有数据。

3) 显示 DEPARTMENTS 表的结构：DESCRIBE DEPARTMENTS;

4)显示 EMPLOYEES 表的结构。创建一个查询， 显示每个雇员的 last name, job_id, hire_date, employee_id. employee_id 显示在第一列， 给 HIRE_DATE 列指定一个别名 STARTDATE。

4) 显示 EMPLOYEES 表的结构，并创建一个查询来显示每个雇员的 last name, job_id, hire_date 和 employee_id。同时，将 hire_date 列的别名设置为 STARTDATE，将 employee_id 列放在第一列：DESCRIBE EMPLOYEES;SELECT employee_id, last_name, job_id, hire_date AS STARTDATEFROM employees;

5)创建一个查询从 EMPLOYEES 表中显示全体在编员工的唯一职务信息。

5) 创建一个查询，显示 EMPLOYEES 表中所有在编员工的唯一职务信息：SELECT DISTINCT job_idFROM employees;

6)创建一个查询，显示收入超过 $12,000 的雇员的名字和薪水。

6) 创建一个查询，显示收入超过 $12,000 的雇员的名字和薪水：

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

7)将 6)的 SQL 语句存到文件 lab2_1.sql 中，运行该脚本执行查询。

7) 将 6)的 SQL 语句存到文件 lab2_1.sql 中，运行该脚本执行查询。

8)创建一个查询，显示雇员号为 176 的雇员的姓名和所在部门的编号。

8) 创建一个查询，显示雇员号为 176 的雇员的姓名和所在部门的编号：

```
SELECT e.last_name, d.department_id

FROM employees e

JOIN departments d ON e.department_id = d.department_id

WHERE e.employee_id = 176;
```

9)修改 lab2_1.sql 文件，显示所有薪水不在 $5000 和 $12000 之间的雇员的名字和薪水。将修改后的 SQL 语句存到文件 lab2_3.sql 中。

9) 修改 lab2_1.sql 文件，显示所有薪水不在 $5000 和 $12000 之间的雇员的名字和薪水。将修改后的 SQL 语句存到文件 lab2_3.sql 中。

```
SELECT last_name, salary

FROM employees

WHERE salary NOT BETWEEN 5000 AND 12000;
```

10) 显示受雇日期在 1998 年 2 月 20 日 和 1998 年 5 月 1 日之间的雇员的名字、岗位和受雇日期，按受雇日期降序排序查询结果。

10) 显示受雇日期在 1998 年 2 月 20 日和 1998 年 5 月 1 日之间的雇员的名字、岗位和受雇日期，按受雇日期降序排序查询结果：

```
SELECT last_name, job_id, hire_date

FROM employees

WHERE hire_date BETWEEN '1998-02-20' AND '1998-05-01'

ORDER BY hire_date DESC;
```

11) 显示所有在部门 20 和 50 中的雇员的名字和部门号，并以名字按字母升序排序。

11) 创建一个查询，显示所有在部门 20 和 50 中的雇员的名字和部门号，并以名字按字母升序排序：

```
SELECT last_name, department_id

FROM employees

WHERE department_id IN (20, 50)

ORDER BY last_name ASC;
```

12）修改 lab2_3.sql 列出收入在 $5,000 和 $12,000 之间，并且在部门 20 或 50 工作的雇员的名字和薪水。将列标题分别显示为 Employee 和 Monthly Salary，将 SQL 保 lab2_6.sql。

12）修改 lab2_3.sql，列出收入在 $5,000 和 $12,000 之间，并且在部门 20 或 50 工作的雇员的名字和薪水。将列标题分别显示为 Employee 和 Monthly Salary，将 SQL 保存到 lab2_6.sql：

```
SELECT last_name AS Employee, salary AS "Monthly Salary"
FROM employees
WHERE salary BETWEEN 5000 AND 12000
AND department_id IN (20, 50);
```

13）显示每一个在 1994 年入职的雇员的名字和入职日期。

13）显示每一个在 1994 年入职的雇员的名字和入职日期：

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date BETWEEN '1994-01-01' AND '1994-12-31';
```

14）显示所有没有主管经理的雇员的名字和工作岗位。

14）显示所有没有主管经理的雇员的名字和工作岗位：

```
SELECT last_name, job_id
FROM employees
WHERE manager_id IS NULL;
```

15）显示所有有佣金的雇员的名字、薪水和佣金，以薪水和佣金的降序排序数据。

15）显示所有有佣金的雇员的名字、薪水和佣金，以薪水和佣金的降序排序数据：

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY salary DESC, commission_pct DESC
```

16）显示所有名字中第三个字母是 a 的雇员的名字。

16）显示所有名字中第三个字母是 a 的雇员的名字：

```
SELECT last_name

FROM employees

WHERE last_name LIKE '__a%';
```

17）显示所有名字中有一个 a 和一个 e 的雇员的名字

17）显示所有名字中有一个 a 和一个 e 的雇员的名字：

```
SELECT last_name

FROM employees

WHERE last_name LIKE '%a%e%';
```

18）显示所有工作是销售代表或者普通职员，并且薪水不等于 $2,500、$3,500 或 $7,000 的雇员的名字、工作和薪水。

18）显示所有工作是销售代表或者普通职员，并且薪水不等于 $2,500、$3,500 或 $7,000 的雇员的名字、工作和薪水：

```
SELECT last_name, job_id, salary

FROM employees

WHERE (job_id = 'SA_REP' OR job_id = 'ST_CLERK')

AND salary NOT IN (2500, 3500, 7000);
```

19）写一个查询显示当前日期，列标签显示为 DATE。

19）写一个查询显示当前日期，列标签显示为 DATE：

```
SELECT CURRENT_DATE AS "DATE";
```

20）对每一个雇员，显示 employee_id, last_name, salary, salary 增加 15%，并且表示成整数，列标签显示为 New Salary，将 SQL 语句存到名为 lab3_2.sql 的文本文件中。

20）对每一个雇员，显示 employee_id, last_name, salary, salary 增加 15%，并且表示成整数，列标签显示为 New Salary，将 SQL 语句存到名为 lab3_2.sql 的文本文件中：

```
SELECT employee_id, last_name, salary, ROUND(salary * 1.15) AS "New Salary"

FROM employees;
```

21）修改查询 lab3_2.sql 添加一个列，该列从新薪水 New Salary 列中减去旧薪水，列标签为 increase，SQL 保存内容到文件 lab3_4.sql 中，运行修订的查询。

21）修改查询 lab3_2.sql，添加一个列，该列为新薪水 New Salary 减去旧薪水，列标签为 increase，将 SQL 保存到文件 lab3_4.sql，并运行修订的查询。

```
SELECT employee_id, last_name, salary, ROUND(salary * 1.15) AS "New Salary",
ROUND((salary * 1.15) - salary) AS increase
FROM employees;
```

22）写一个查询用首写字母大写，其它字母小写显示雇员的 last_name， 并显示姓名的长度，对所有姓名开始字母是 J、A 或 M 的雇员， 给每列一个适当的标签， 用 last_name 对结果排序。

22）写一个查询，将雇员的 last_name 以首字母大写、其余字母小写的形式显示，并显示姓名的长度。查询结果仅包括以 J、A 或 M 开头的雇员的 last_name，并对结果按照 last_name 进行排序。

```
SELECT UPPER(SUBSTRING(last_name, 1, 1)) + LOWER(SUBSTRING(last_name, 2)) AS "Last Name",
LENGTH(last_name) AS "Name Length"
FROM employees
WHERE last_name LIKE 'J%' OR last_name LIKE 'A%' OR last_name LIKE 'M%'
ORDER BY last_name;
```

23）对每一个雇员， 显示其 last_name，并计算从雇员受雇日期到今天的月数， 列标签为 MONTHS__WORKED，按受雇月数排序结果，四舍五入月数到最靠近的整数月 。

23）对于每个雇员，显示其 last_name，并计算从雇员受雇日期到今天的月数，列标签为 MONTHS_WORKED。按照受雇月数对结果进行排序，四舍五入月数到最接近的整数月。

```
SELECT last_name, ROUND(DATEDIFF(CURRENT_DATE, hire_date) / 30) AS "MONTHS_WORKED"
FROM employees
ORDER BY MONTHS_WORKED;
```

24）创建一个查询显示所有雇员的 last_name 和 salary. salary 格式化为 15 个字符长度， 用$左填充，列标签为 SALARY。

24）创建一个查询，显示所有雇员的 last_name 和 salary。将 salary 格式化为 15 个字符长度，用 $ 左对齐填充，列标签为 SALARY。

```
SELECT last_name, LPAD(CONCAT('$', salary), 15, '$') AS "SALARY"
FROM employees;
```

25）显示每一个雇员的 last_name，hire_date 和雇员入职周内天数，列标签 DAY，用星期一作 为每周的起始日。

25）显示每个雇员的 last_name、hire_date 和雇员入职周内的天数，列标签为 DAY，以星期一作为每周的起始日。

```
SELECT last_name, hire_date, (DAYOFWEEK(hire_date) + 5) % 7 + 1 AS "DAY"
FROM employees;
```

26）创建一个查询显示雇员的 last_names 和 commission(佣金)比率。如果雇员没有佣金显示“No Commission”，列标签为 COMM。

26）创建一个查询，显示雇员的 last_name 和 commission 比率。如果雇员没有佣金，显示 "No Commission"，列标签为 COMM。

```
SELECT last_name,
       CASE
           WHEN commission_pct IS NULL THEN 'No Commission'
           ELSE CONCAT(commission_pct * 100, '%')
       END AS "COMM"
FROM employees;
```

27）显示所有雇员的最高、最低、合计和平均薪水，列标签分别为： maximum、minmum、sum、average。四舍五入结果为最近的整数。

27）显示所有雇员的最高、最低、合计和平均薪水，列标签分别为：maximum、minimum、sum、average。四舍五入结果为最近的整数。

```
SELECT MAX(salary) AS "maximum",
       MIN(salary) AS "minimum",
       SUM(salary) AS "sum",
       ROUND(AVG(salary)) AS "average"
FROM employees;
```

28）对 27） 的问题显示每种工作类型的最低、最高、合计、平均薪水。

28）对于问题 27），显示每种工作类型的最低、最高、合计、平均薪水。

```
SELECT job_id,
       MAX(salary) AS "maximum",
       MIN(salary) AS "minimum",
       SUM(salary) AS "sum",
       ROUND(AVG(salary)) AS "average"
FROM employees
GROUP BY job_id;
```

29）显示每一个工作岗位的人数。

29）显示每个工作岗位的人数。

```
SELECT job_id, COUNT(*) AS "Number of Employees"
FROM employees
GROUP BY job_id;
```

30）显示所有经理人数。

30）显示所有经理的人数。

```
SELECT COUNT(*) AS "Number of Managers"
FROM employees
WHERE job_id = 'IT_PROG' OR job_id = 'ST_MAN';
```

31）显示最高与最低薪水之间的差额。

31）显示最高与最低薪水之间的差额。

```
SELECT MAX(salary) - MIN(salary) AS "Salary Difference"
FROM employees;
```

32）显示每个部门的名字、地点、人数和部门中所有雇员的平均薪水。四舍五入薪水到两位小数。

32）显示每个部门的名字、地点、人数和部门中所有雇员的平均薪水。四舍五入薪水到两位小数。

```
SELECT d.department_name, d.location_id, COUNT(e.employee_id) AS "Number of Employees",
ROUND(AVG(e.salary), 2) AS "Average Salary"
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name, d.location_id;
```

33）写一个查询显示所有雇员的 last_name, department_id and department_name。

33）写一个查询显示所有雇员的 last_name, department_id 和 department_name。

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

34） 创建一个在部门 80 中的所有工作岗位的唯一列表，在输出中包括部门的地点。

34）创建一个在部门 80 中的所有工作岗位的唯一列表，在输出中包括部门的地点。

```
SELECT DISTINCT job_id, location_id
FROM employees
WHERE department_id = 80;
```

35）显示所有在其 last_name 中有一个小写 a 的雇员的 last_name 和 department_name。

35）显示所有在其 last_name 中有一个小写 a 的雇员的 last_name 和 department_name。

```
SELECT last_name, department_name
FROM employees
JOIN departments ON employees.department_id = departments.department_id
WHERE last_name LIKE '%a%';
```

36) 写一个查询显示那些工作在 Toronto 的所有雇员的 last_name、job_id、department_id 和 department_name。

36) 写一个查询显示那些工作在 Toronto 的所有雇员的 last_name、job_id、department_id 和 department_name。

```
SELECT e.last_name, e.job_id, e.department_id, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id
WHERE d.location_id = 'Toronto';
```

37) 显示雇员的 last_name 和 employee_id 连同他们的经理的 last_name 和 manager_id， 列标签为 employee、emp_id、manager 和 mgr_id。

37) 显示雇员的 last_name 和 employee_id 连同他们的经理的 last_name 和 manager_id，列标签为 employee、emp_id、manager 和 mgr_id。

```
SELECT e.last_name AS "employee", e.employee_id AS "emp_id", m.last_name AS "manager",
e.manager_id AS "mgr_id"
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;
```

38) 创建一个查询， 显示所有与被指定雇员工作在同一部门的雇员的 last_names、department_id。给每列一个适当的标签。

38) 创建一个查询，显示所有与被指定雇员工作在同一部门的雇员的 last_names、department_id。给每列一个适当的标签。

```
SELECT e.last_name AS "last_names", e.department_id
FROM employees e
JOIN employees m ON e.department_id = m.department_id
WHERE m.last_name = '指定雇员的last_name';
```

39) 写一个查询显示与 zlotkey 在同一部门的雇员的 last name 和 hire date，结果中不包括 zlotkey。

39) 写一个查询显示与 zlotkey 在同一部门的雇员的 last name 和 hire date，结果中不包括 zlotkey。

```
SELECT last_name, hire_date
FROM employees
WHERE department_id = (SELECT department_id FROM employees WHERE last_name = 'zlotkey')
AND last_name != 'zlotkey';
```

40) 创建一个查询显示所有薪水高于平均薪水的雇员的雇员编号和名字，按薪水的升序排序。

40) 创建一个查询显示所有薪水高于平均薪水的雇员的雇员编号和名字，按薪水的升序排序。

```
SELECT employee_id, last_name
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
ORDER BY salary ASC;
```

41) 显示所有部门位置编号（location_id）是 1700 的雇员的 last name、departtment_id 和 job_id。

41) 显示所有部门位置编号（location_id）是 1700 的雇员的 last name、department_id 和 job_id。

```
SELECT last_name, department_id, job_id
FROM employees
WHERE department_id IN (SELECT department_id FROM departments WHERE location_id = 1700);
```

42) 显示在 Executive 部门的每个雇员的 department_id、last_name 和 job_id。

42) 显示在 Executive 部门的每个雇员的 department_id、last_name 和 job_id。

```
SELECT department_id, last_name, job_id
FROM employees
WHERE department_id = (SELECT department_id FROM departments WHERE department_name =
'Executive');
```

THE END