## SQL Exercises (with answers)

Give the SQL commands for the following and answer any associated questions:

# (A) "Simple" SELECT Command Questions

1. Display all information in the tables EMP and DEPT.

```
SELECT * FROM emp;
SELECT * FROM dept;
```

2. Display only the hire date and employee name for each employee.

```
SELECT hiredate, ename FROM emp;
```

3. Display the hire date, name and department number for all clerks.

```
SELECT hiredate, ename, deptno FROM emp WHERE job = 'CLERK';
```

4. Display the names and salaries of all employees with a salary greater than 2000.

```
SELECT ename, sal FROM emp WHERE sal > 2000;
```

5. Display the names of all employees with an 'A' in their name.

```
SELECT ename FROM emp WHERE ename LIKE '%A%';
```

6. Display the names of all employees with exactly 5 letters in their name.

```
SELECT ename FROM emp WHERE ename LIKE '____';
```

7. Display the names and hire dates of all employees hired in 1981 or 1982 (Note in Visual Basic or Access SQL you need to refer to dates in a WHERE clause between #s, eg. #1 Jan 2000#).

```
SELECT ename, hiredate FROM emp
WHERE hiredate LIKE '%1981' OR hiredate LIKE '%1982';
—OR—
```

```
{\tt SELECT\ ename,\ hiredate\ FROM\ emp}
```

```
WHERE hiredate >= '1/1/1981' AND hiredate <= '31/12/1982';
```

8. Display the names and dates of employees with the column headers "Name" and "Start Date"

```
SELECT ename AS "Name", hiredate AS "Start Date" FROM emp;
```

9. Display the names and hire dates of all employees in the order they were hired.

```
SELECT ename, hiredate FROM emp ORDER BY hiredate;
```

10. Display the names and salaries of all employees in reverse salary order.

```
SELECT ename, sal FROM emp ORDER BY sal DESC;
```

11. Display 'ename of department deptno earned commission \$' for each salesman in reverse salary order.

```
SELECT ename || ' of department ' || deptno || ' earned commission $' || comm FROM emp WHERE job = 'SALESMAN' ORDER BY sal DESC;
```

12. Display the department numbers of all departments employing a clerk.

```
SELECT DISTINCT deptno FROM emp WHERE emp.job = 'CLERK';
```

## (B) Group SELECT Command Questions

1. Display the maximum, minimum and average salary and commission earned.

```
SELECT max(sal), min(sal), avg(sal), max(comm), min(comm), avg(comm) FROM emp;
```

2. Display the department number, total salary payout and total commission payout for each department.

```
SELECT deptno, sum(sal), sum(comm) FROM emp GROUP BY deptno;
```

3. Display the department number, total salary payout and total commission payout for each department that pays at least one employee commission.

```
SELECT deptno, sum(sal), sum(comm) FROM emp GROUP BY deptno HAVING sum(comm) > 0;
```

- 4. Display the department number and number of clerks in each department.
  - SELECT deptno, count(job) FROM emp WHERE job = 'CLERK' GROUP BY deptno;
- 5. Display the department number and total salary of employees in each department that employs four or more people.
  - SELECT deptno, sum(sal) FROM emp GROUP BY deptno HAVING count(empno) >= 4;
- 6. Display the employee number of each employee who manages other employees with the number of people he or she manages.
  - SELECT mgr, count(mgr) FROM emp WHERE mgr IS NOT NULL GROUP BY mgr;

## (C) Join SELECT Command Questions

- 1. Display the name of each employee with his department name.
  - SELECT ename, dname FROM emp INNER JOIN dept ON emp.deptno = dept.deptno;
- 2. Display a list of all departments with the employees in each department.
  - SELECT dname, ename FROM dept LEFT OUTER JOIN emp ON dept.deptno = emp.deptno;
- 3. Display all the departments with the manager for that department.
  - SELECT dname, ename FROM emp INNER JOIN dept ON emp.deptno = dept.deptno WHERE job = 'MANAGER';
- 4. Display the names of each employee with the name of his/her boss.
  - SELECT s.ename, b.ename FROM emp s INNER JOIN emp b ON s.mgr = b.empno;
- 5. Display the names of each employee with the name of his/her boss with a blank for the boss of the president.
  - SELECT s.ename, b.ename FROM emp s LEFT OUTER JOIN emp b ON s.mgr = b.empno;
- 6. Display the employee number and name of each employee who manages other employees with the number of people he or she manages.

```
SELECT a.mgr, b.ename, count(a.mgr) FROM emp a INNER JOIN emp b
ON a.mgr = b.empno WHERE a.mgr IS NOT NULL GROUP BY a.mgr, b.ename;
```

7. Repeat the display for the last question, but this time display the rows in descending order of the number of employees managed.

```
SELECT a.mgr, b.ename, count(a.mgr) FROM emp a INNER JOIN emp b
ON a.mgr = b.empno WHERE a.mgr IS NOT NULL GROUP BY a.mgr, b.ename
ORDER BY count(a.mgr) DESC;
```

## (D) SELECT with Subqueries Questions

1. Display the names and job titles of all employees with the same job as Jones.

```
SELECT ename, job FROM emp
WHERE job = (SELECT job FROM emp WHERE ename = 'JONES');
```

2. Display the names and department name of all employees working in the same city as Jones.

```
SELECT ename, dname FROM emp INNER JOIN dept ON emp.deptno = DEPT.deptno WHERE loc = (SELECT loc FROM emp INNER JOIN dept ON emp.deptno = DEPT.deptno WHERE ename = 'JONES');
```

3. Display the name of the employee whose salary is the lowest.

```
SELECT ename FROM emp WHERE sal = (SELECT min(sal) FROM emp);
```

4. Display the names of all employees except the lowest paid.

```
SELECT ename FROM emp WHERE sal > (SELECT min(sal) FROM emp);
```

5. Display the names of all employees whose job title is the same as anyone in the sales dept.

```
SELECT ename FROM emp WHERE job IN (SELECT DISTINCT job FROM emp INNER JOIN dept ON emp.deptno = DEPT.deptno WHERE dname = 'SALES');
```

6. Display the names of all employees who work in a department that employs an analyst.

```
SELECT ename FROM emp WHERE deptno IN (SELECT DISTINCT emp.deptno FROM emp INNER JOIN dept ON emp.deptno = dept.deptno WHERE job = 'ANALYST');
```

7. Display the names of all employees with their job title, their current salary and their salary following a 10% pay rise for clerks and a 7% pay rise for all other employees.

```
SELECT ename, job, sal, 1.1 * sal AS "newsal" FROM emp WHERE job = 'CLERK' UNION SELECT ename, job, sal, 1.07 * sal AS "newsal" FROM emp WHERE job <> 'CLERK';
```

8. Display the names of all employees with their salary and commission earned. Employees with a null commission field should have 0 in the commission column.

```
SELECT ename, sal, comm FROM emp WHERE comm IS NOT NULL UNION SELECT ename, sal, O FROM emp WHERE comm IS NULL;
```

9. Display the names of ALL employees with the total they have earned (ie. salary plus commission).

```
SELECT ename, sal + comm AS "earnings" FROM emp WHERE comm IS NOT NULL UNION SELECT ename, sal FROM emp WHERE comm IS NULL;
```

10. Repeat the display for the last question but this time display in descending order of earnings.

```
SELECT ename, sal + comm AS "earnings" FROM emp WHERE comm IS NOT NULL UNION SELECT ename, sal FROM emp WHERE comm IS NULL ORDER BY earnings DESC;
```

#### (E) Creating and Modifying Tables

1. Add a new Department to the DEPT table, and add a Manager and two Clerks to the EMP table that will belong to the new department.

```
INSERT INTO dept VALUES (50, 'NEWDEPT', 'LONDON');
INSERT INTO emp
    VALUES (8001, 'FRED', 'MANAGER', 7839, '14/01/1984', 3100, null, 50);
INSERT INTO emp
    VALUES (8002, 'JIM', 'CLERK', 8001, '18/04/1984', 1020, null, 50);
INSERT INTO emp
    VALUES (8003, 'SHEILA', 'CLERK', 8001, '08/12/1984', 955, null, 50);
```

2. Transfer one of the new clerks to a different department and transfer one of the previously existing clerks to your new department.

```
UPDATE emp SET deptno = 40, mgr = 7788 WHERE empno = 8002;
UPDATE emp SET deptno = 50, mgr = 8001 WHERE empno = 7876;
```

3. Create a new table called JOBS with two fields, a SMALLINT called JOBNO and a 15 character text field called JOB.

```
CREATE TABLE jobs ( jobno SMALLINT, job VARCHAR(15) );
```

4. Fill your new JOBS table with null values for the JOBNO and the job values from the EMP table. There should be only one row with each job type (ie. no repeats).

```
INSERT INTO jobs (job) SELECT DISTINCT job FROM emp;
```

5. Give a unique job number to each job type.

```
UPDATE jobs SET jobno = 10 WHERE job = 'ANALYST';

UPDATE jobs SET jobno = 20 WHERE job = 'CLERK';

UPDATE jobs SET jobno = 30 WHERE job = 'MANAGER';

UPDATE jobs SET jobno = 40 WHERE job = 'SALESMAN';

UPDATE jobs SET jobno = 50 WHERE job = 'PRESIDENT';
```

6. Create a new empty table called EMP1. This table should have the same fields as EMP but with an additional field called JOBNO of type SMALLINT. (Note—also make EMPNO and DEPTNO type SMALLINT.)

```
CREATE TABLE "emp1" (
    "empno" SMALLINT,
    "ename" VARCHAR(15),
    "job" VARCHAR(15),
    "mgr" SMALLINT,
    "hiredate" DATE,
    "sal" SMALLINT,
    "comm" SMALLINT,
    "deptno" SMALLINT,
    "jobno" SMALLINT);
```

7. Fill your new EMP1 table with the data from EMP and JOBS.

```
INSERT INTO emp1 SELECT emp.*, jobno FROM emp INNER JOIN jobs
   ON emp.job = jobs.job;
```

8. Remove the JOB column from your EMP1 table.

```
ALTER TABLE emp1 DROP COLUMN job;

—OR—

SELECT empno, ename, mgr, hiredate, sal, comm, deptno, jobno
    INTO temp FROM emp1;

DROP TABLE emp1;

ALTER TABLE temp RENAME TO emp1;
```

9. Display the data from the EMP1 and JOBS tables so that the output is identical to the original EMP table.

```
SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM emp1 INNER JOIN jobs ON emp1.jobno = jobs.jobno;
```

### (F) Transactions

Add a new row to a table, delete a row from a table and modify a row in a table. Make the changes
to more than one table. Now enter the command ROLLBACK. What has happened to your changes?
They have all been lost

[this answer may be wrong depending on your SQL program]

2. Produce a further set of inserts, deletes and modifications to the tables, enter the command COMMIT and then produce yet more changes to the data. Now enter the command ROLLBACK. What has happened to your changes?

They have all been lost back to the COMMIT command [this answer may be wrong depending on your SQL program]

3. Produce a further set of inserts, deletes and modifications to the tables, then create a new table called TEMP and then produce yet more changes to the data. Now enter the command ROLLBACK. What has happened to your changes?

They have all been lost back to the creation of the TEMP table [this answer may be wrong depending on your SQL program]

4. Produce a further set of inserts, deletes and modifications to the tables, and then exit the SQL program. Re-start the SQL program and produce yet more changes to the data. Now enter the command ROLLBACK. What has happened to your changes?

Changes have been made before exiting the program, but changes made after restarting it have been lost [this answer may be wrong depending on your SQL program]

### (G) Views

1. Create a view called VIEWEMP which gives the same displays as EMP but is based on the EMP1 and JOBS tables.

```
CREATE VIEW viewemp AS SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM emp1 INNER JOIN jobs ON emp1.jobno = jobs.jobno;
```

2. Create a view called DEPTSUM with two columns called DEPARTMENT and SUMSAL containing the name of each department and the sum of the salaries for all employees in the department. Look at it using a SELECT command. Now alter one of the salaries in the EMP table using the UPDATE command—is the DEPTSUM affected?

CREATE VIEW deptsum AS SELECT dname, SUM(sal) FROM emp INNER JOIN dept ON emp.deptno = dept.deptno GROUP BY dname;

SELECT \* FROM deptsum;

UPDATE emp SET sal = 1000 WHERE empno = 7369;

SELECT \* FROM deptsum;

DEPTSUM has been updated

3. Create a view called BOSS which has the name and number of each employee with the name and number of his or her manager (with blanks alongside any employee that has no manager). Give each column in the view a suitable name. Change one of the entries in the EMP table to give an employee a different manager and check this is reflected in the BOSS view.

CREATE VIEW boss AS SELECT a.ename AS ename, a.empno AS empno,

b.ename AS bname, b.empno AS bossno

FROM emp a LEFT OUTER JOIN emp b ON a.mgr = b.empno;

SELECT \* FROM boss;

UPDATE emp SET mgr=7566 WHERE empno = 7934;

SELECT \* FROM boss;

4. Delete the DEPTSUM and BOSS views.

DROP VIEW deptsum;

DROP VIEW boss;

5. Create a view called SALES1 which has all the columns of the EMP table except the salary and commission columns, and with only the rows corresponding to employees in department number 30.

```
CREATE VIEW sales1 AS SELECT empno, ename, job, mgr, hiredate, deptno FROM emp WHERE deptno = 30;
```

6. Create a view called SALES2 which has the same columns as SALES1 except the department name is included instead of the department number. Do this by basing the new view on a join of the SALES1 view joined to the DEPT table.

```
CREATE VIEW sales2 AS SELECT empno, ename, job, mgr, hiredate, dname FROM sales1 INNER JOIN dept ON sales1.deptno = dept.deptno;
```

7. Insert a new employee into the SALES1 view with the INSERT command. Look in the EMP table and SALES1 and SALES2 views to check it is there.

INSERT INTO sales1

```
VALUES (8101, 'BOB', 'ANALYST', 7839, '09/06/1982', 20);
```

This will not work in PostgreSQL without extra commands

8. Insert a new employee into the SALES2 view with the INSERT command. Can this be done?

INSERT INTO sales2

```
VALUES (8101, 'BOB', 'ANALYST', 7839, '09/06/1982', 20);
```

This will not work in PostgreSQL without extra commands

9. Alter the view SALES1 so that it has an additional column called INCOME with the sum of the salary and commission for each employee. (Is it really possible to alter a view?)

```
DROP VIEW sales1;
```

DROP VIEW sales1 CASCADE;

CREATE VIEW sales1 AS

SELECT empno, ename, job, mgr, hiredate, deptno, sal + comm AS income FROM emp WHERE comm IS NOT null AND deptno = 30 UNION SELECT empno, ename, job, mgr, hiredate, deptno, sal AS income FROM emp WHERE comm IS null AND deptno = 30;

```
SELECT * FROM sales1;
```

It is not possible to alter a view, only to drop it and re-create it

10. Now the SALES1 view has the extra INCOME column is it possible to insert another employee to the department using this view?

```
INSERT INTO sales1
   VALUES (8101, 'BOB', 'ANALYST', 7839, '09/06/1982', 20);
```

This will not work in PostgreSQL without extra commands

## (H) Table Constraints

- 1. Create a table called CLIENTS with the following fields:
  - $\rightarrow$  a text field of 20 characters called CNAME with the clients name
  - ightarrow an integer field called EMPNO which is the employee ID of the associated sales rep

The table should be configured such that:

- $\rightarrow$  blank fields cannot be inserted into the CNAME column
- $\rightarrow$  if the EMPNO field is not filled it is automatically given a value of 7654
- $\rightarrow$  the EMPNO field cannot be given a value outside the range 7000 to 8000

```
CREATE TABLE clients (
    cname VARCHAR(20) NOT NULL,
    empno SMALLINT DEFAULT 7654,
    CHECK (cname <> ''),
    CHECK (empno >= 7000 AND empno <= 8000));</pre>
```

2. What happens if an attempt is made to insert a row into the CLIENTS table with a empty CNAME field?

```
INSERT INTO clients (cname, empno) VALUES ('', 7777);
INSERT INTO clients (empno) VALUES (7777);
```

It will not work due to the check constraints

3. What happens if an attempt is made to insert a row into the CLIENTS table with a empty EMPNO field?

```
INSERT INTO clients (cname) VALUES ('DARCY');
```

The row is given an EMPNO of 7654

4. What happens if an attempt is made to insert a row into the CLIENTS table with a value of 6789 in the EMPNO field?

```
INSERT INTO clients (cname, empno) VALUES ('JANE', 6789);
```

It will not work due to the check constraints

5. Create an index on the ENAME field in the EMP table.

```
CREATE INDEX emp_ename_index ON emp (ename);
```

6. What is now different when you list all rows in the EMP table with a simple SELECT statement?

```
SELECT * FROM emp;
```

There is no change (with some databases, the ENAME column would now be sorted)

7. Create a unique index on the EMPNO field in the EMP table.

```
CREATE UNIQUE INDEX emp_empno_index ON emp (empno);
```

8. What is now different when you list all rows in the EMP table with a simple SELECT statement?

```
SELECT * FROM emp;
```

There is no change (with some databases, the EMPNO column would now be sorted because of the unique index)

9. Delete the unique index on the EMPNO field in the EMP table and check that listing the table now gives the same output as before the index was created.

DROP INDEX emp\_empno\_index;

```
SELECT * FROM emp;
```

10. What happens when you attempt to create a unique index on the MGR field in the EMP table? (Check to see if the index has been created by attempting to delete the index)

```
CREATE UNIQUE INDEX emp_mgr_index ON emp (mgr);
```

The unique index cannot be created because the MGR column has duplicated entries

11. Change the EMP table so that the EMPNO field is specified as the primary key.

```
ALTER TABLE emp ALTER COLUMN empno SET NOT NULL;
```

```
ALTER TABLE emp ADD CONSTRAINT emp_empno_pkey PRIMARY KEY (empno);
```

12. What is now different when you list all rows in the EMP table with a simple SELECT statement?

```
SELECT * FROM emp;
```

There is no change (with some databases, the EMPNO column would now be sorted because of the unique index)

13. What happens if you try and insert an extra row in the EMP table with a blank EMPNO field?

```
INSERT INTO EMP
```

```
VALUES (null, 'BOB', 'ANALYST', 7839, '09/06/1982', 1000, null, 20);
INSERT INTO EMP (ename, job, mgr, hiredate, sal, comm, deptno)
VALUES ('BOB', 'ANALYST', 7839, '09/06/1982', 1000, null, 20);
```

Can't insert because of the null EMPNO

14. What happens if you try and insert an extra row in the EMP table with the value of the EMPNO field the same as in an existing row in the table?

```
INSERT INTO EMP (empno, ename, job, mgr, hiredate, sal, comm, deptno)
    VALUES (7499, 'BOB', 'ANALYST', 7839, '09/06/1982', 1000, null, 20);
```

Can't insert because of the duplicate  ${\sf EMPNO}$  value

15. Change the EMP table so that the EMPNO field is no longer specified as the primary key.

```
ALTER TABLE emp DROP CONSTRAINT emp_empno_pkey;
```

ALTER TABLE emp ALTER COLUMN empno DROP NOT NULL;