1a. Write a simple query to display the name, job, hire date and employee number of each employee from the emp table.

```
SELECT ENAME, JOB, HIREDATE, EMPNO

FROM EMP;
```

1b. Now, display all employees in the emp table including all columns.

```
SELECT * FROM EMP;
```

1c. Now rewrite the query to display names of all employees, making sure that two employees with same name don't appear twice in the result set.

```
SELECT distinct ENAME FROM EMP;
```

2. Write a query to add a record to the emp table:

```
INSERT INTO EMP VALUES (1456, 'John Smith', 'Analyst', 7566,'2002-01-01', 2000, NULL, 20);
```

EMPNO: 1456

ENAME: John Smith

JOB: Analyst

SAL: 2000

HIREDATE: 1/1/02

COMM:

DEPTNO: 20

MGR: 7566

3. Show how will you modify the above record to reflect SAL = 3000

```
UPDATE EMP

SET SAL = 3000

WHERE EMPNO=1456;
```

4a. Write a SQL statement to show the employee number, employee name, hiredate of employees where name has 2 L's.

```
SELECT EMPNO, ENAME, HIREDATE

FROM EMP

WHERE ENAME Like '%L%L%';
```

4b. Now from the above result set, display the same fields, for employees whose names end with "N".

```
SELECT EMPNO, ENAME, HIREDATE

FROM (SELECT EMPNO, ENAME, HIREDATE

FROM EMP

WHERE ENAME Like '%L%L%') as A

WHERE A.ENAME LIKE '%N';
```

5. Display all the fields of dept table, where location is BOSTON.

```
SELECT *

FROM dept

WHERE LOC = 'BOSTON';
```

6. Display employee number, employee name, department number, job for an employee who is not a manager and is not a clerk in department number 10.

```
SELECT EMPNO, ENAME, DEPTNO, JOB

FROM emp

WHERE JOB <> 'manager' and JOB<>'clerk' and DEPTNO = 10;
```

7. Display all employees whose commission is greater than zero and salary is between 1000 and 3000.

```
SELECT *

FROM emp

WHERE COMM > 0 and SAL >=1000 and SAL <=3000;
```

8. Write a SQL statement to show employees, who don't have any managers.

```
SELECT *

FROM emp

WHERE MGR is null;
```

9. Write a SQL statement to display employee number, employee name, salary, manager for all employees, whose managers have employee numbers 7566, 7788.

```
SELECT EMPNO, ENAME, SAL, MGR

FROM emp

WHERE MGR = 7566 or MGR = 7788;
```

10. Write a query to display employee number, employee name, hiredate, manager's name for those employees, whose manager's name starts with K or M or S. Label the columns Employee Number, Employee Name, Hiredate, Mgr Name.

```
SELECT emp.EMPNO as Employee_Number, emp.ENAME as Employee_Name, emp.HIREDATE as Hiredate, M.ENAME as Mgr_Name

FROM (SELECT EMPNO, ENAME FROM emp WHERE ENAME Like 'K%' or ENAME Like 'M%' or ENAME Like 'S%') as M, emp

WHERE emp.MGR = M.EMPNO;
```

11. Create a query that will display the employee name, department number, department name and all the employees that work in the same department as a given employee. Give each column an appropriate label.

```
SELECT emp.ENAME as employee_name, emp.DEPTNO as department_no, dept.DNAME as department_name

FROM emp, (SELECT DEPTNO FROM emp WHERE ENAME = 'KING') as dno, dept

WHERE emp.DEPTNO = dno.DEPTNO and dept.DEPTNO=dno.deptno;
```

12. Write a query to display the department name, location of all employees who are clerks.

```
SELECT dept.DNAME, dept.LOC

FROM (SELECT * FROM emp WHERE JOB = 'CLERK') as clerk, dept

WHERE clerk.DEPTNO = dept.DEPTNO;
```

13. Insert a new row into the department table: department number = 50, department name = training, location = San Francisco. Now create a query to display all the employees in department number 20 and 50. Columns to be displayed are emp number, emp name, dept name, dept location.

```
INSERT INTO dept VALUES (50, 'training', 'San Francisco');

SELECT emp.EMPNO, emp.ENAME, dept.DNAME, dept.LOC, dept.DEPTNO

FROM emp, dept

WHERE emp.DEPTNO = dept.DEPTNO and (emp.DEPTNO = 20 or emp.DEPTNO = 50);
```

14. Insert a new row into the emp table - you can choose any values for the fields, but department number should be null. Now create a query to display all the employees and all the departments, using joins.

```
INSERT INTO emp VALUES (6666, 'HAHA', 'SALESMAN', 7839, '2010-12-12', 4000, NULL, 50);

SELECT *

FROM emp

INNER JOIN dept

ON emp.DEPTNO = dept.DEPTNO;
```

15. Display the manager number and the salary of the lowest paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is less than $1000. Sort the output in descending order of salary.

```
SELECT *

FROM (SELECT MGR, MIN(SAL) as ms

FROM emp

WHERE MGR is not null

GROUP BY MGR) as M

WHERE ms >= 1000 ORDER BY ms DESC;
```

16. Write a query to display the department name, location name, number of employees, and the average salary for all employees in that department. Label the columns dname, loc, Number of People, and Salary, respectively.

```
SELECT DNAME, LOC, COUNT(empno), avg(sal)

from dept, emp

where dept.deptno=emp.DEPTNO
```

```
group by emp.DEPTNO
```

17. Write a query to display department names with salary grade, minimum salary and average commission. For departments with null commission, you should display 0. (salgrade table can be used for getting salary grade).

```
SELECT dname dName,loc location, numberofpeople 'Number of People', minsal 'Lowest Salary', ifnull(avgcommission, 0) 'Average Commission', salgrade.grade 'Salary Grade'

FROM

(SELECT DEPT.DNAME 'dname', dept.loc 'loc', COUNT(emp.EMPNO) NumberofPeople, ROUND(AVG(sal), 2) Salaryavg, round(AvG(Comm), 2) AvgCommission, round(min(sal), 2) Minsal

FROM emp

RIGHT JOIN dept ON emp.DEPTNO = dept.DEPTNO

GROUP BY dept.DNAME , dept.DEPTNO) m

left join

salgrade ON m.salaryavg between salgrade.losal and salgrade.hisal
```

18. What is difference between COUNT(*), COUNT(col_name), COUNT(DISTINCT(col_name)), COUNT(ALL(col_name))? Explain with examples.

```
COUNT(*): Count the number of rows in table;

eg: SELECT COUNT(*) FROM EMP;

COUNT(col_name): Only count the number of rows have non-null value in the col_name.

eg: SELECT COUNT(DEPTNO) FROM EMP;

COUNT(DISTINCT(col_name)): Count the number rows holding different value in col_name;

eg: SELECT COUNT(DISTINCT(DEPTNO)) FROM EMP;

COUNT(ALL(col_name)): Same as COUNT(*)
```

eg: SELECT COUNT(ALL(DEPTNO)) FROM EMP;

19. Display the employee number, name, salary, and salary increase by 15% expressed as a whole number. Label the column New Salary.

SELECT EMPNO,ENAME,SAL,1.15*SAL AS 'NEW SALARY' FROM EMP;

20. Create a query that displays the employees names and indicates the amounts of their salaries through asterisks. Each asterisk signifies hundred dollars. Sort the data in descending order of salary. Label the column EMPLOYEE_AND_THEIR_SALARIES.

SELECT ENAME, SAL/100 AS EMPLOYEE_AND_THEIR_SALARIES FROM EMP ORDER BY SAL DESC;

21. Display the employees name, username, hire date, salary and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format mm/dd/yy. Salary should be rounded. Username is first two letters of the name

in the lower case.

22. Use subquery to display all employees, in department location 'BOSTON' with a salary of greater than $1000.

SELECT * FROM (SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,

DEPT.DEPTNO, DEPT.DNAME,DEPT.LOC FROM EMP INNER JOIN DEPT ON

EMP.DEPTNO=DEPT.DEPTNO) AS OK WHERE SAL>1000 AND LOC=‘BOSTON';

23. Write a query to display the employee name, job, and hire date for all employees who started between 01/01/81 to 12/31/81. Concatenate the name and job together, separated by a space and comma, and label the column Employees.

SELECT CONCAT_WS(' ,',ENAME,JOB) AS EMPLOYEES,HIREDATE FROM EMP WHERE

```
HIREDATE BETWEEN '1981-01-01' AND '1981-12-31';
```

24. Explain the usage of correlated subqueries, with an example.

A correlated subquery is a subquery (a query nested inside another query) that uses values from outer query. The subquery is evaluated once for each row processed by the outer query.

Example correlated subquery:

```
SELECT p.product_name FROM product p

WHERE p.product_id = (SELECT o.product_id FROM order_items o

WHERE o.product_id = p.product_id);
```

An inline view is a SELECT statement in the FROM-clause of another SELECT statement. In-line views are commonly used to simplify complex queries by removing join operations and condensing several separate queries into a single query.

Example inline view:

```
SELECT *

 FROM ( SELECT deptno, count(*) emp_count

       FROM emp

       GROUP BY deptno ) emp,

     dept

 WHERE dept.deptno = emp.deptno;
```