Good SQL Queries Collection

```
/* CREATE TABLE FOR DEPARTMENT */
CREATE TABLE DEPARTMENT
(
“DEPT_ID” NUMBER,
“DEPT_NAME” VARCHAR2(30),
PRIMARY KEY (“DEPT_ID”)
)

/* CREATE TABLE FOR EMP */
CREATE TABLE EMPLOYEE
(
“EMP_ID” NUMBER,
“MGR_ID” NUMBER,
“DEPT_ID” NUMBER,
“EMP_NAME” VARCHAR2(30),
“SAL” NUMBER,
“DOJ” DATE,
PRIMARY KEY (“EMP_ID”) ENABLE,
FOREIGN KEY (“MGR_ID”) REFERENCES EMPLOYEE (“EMP_ID”) ENABLE,
FOREIGN KEY (“DEPT_ID”) REFERENCES DEPARTMENT (“DEPT_ID”) ENABLE
)

/* INSERT STATEMENT FOR DEPARTMENT */
INSERT INTO DEPARTMENT values (1,’HR’);
INSERT INTO DEPARTMENT values (2,’Engineering’);
INSERT INTO DEPARTMENT values (3,’Marketing’);
INSERT INTO DEPARTMENT values (4,’Sales’);
INSERT INTO DEPARTMENT values (5,’Logistics’);

/* INSERT STATEMENT FOR EMPLOYEE */
INSERT INTO EMPLOYEE values (1, NULL, 2,’Hash’, 100, to_date(‘2012-01-01’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (2, 1, 2, ‘Robo’, 100, to_date(‘2012-01-01’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (3, 2, 1, ‘Privy’, 50, to_date(‘2012-05-01’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (4, 1, 1, ‘Inno’, 50, to_date(‘2012-05-01’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (5, 2, 2, ‘Anno’, 80, to_date(‘2012-02-01’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (6, 1, 2, ‘Darl’, 80, to_date(‘2012-02-11’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (7, 1, 3, ‘Pete’, 70, to_date(‘2012-04-16’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (8, 7, 3, ‘Meme’, 60, to_date(‘2012-07-26’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (9, 2, 4, ‘Tomiti’, 70, to_date(‘2012-07-07’, ‘YYYY-MM-DD’));
INSERT INTO EMPLOYEE values (10, 9, 4, ‘Bhuti’, 60, to_date(‘2012-08-24’, ‘YYYY-MM-DD’));

SELECT e.EMP_NAME,e.EMP_ID,d.DEPT_NAME,d.DEPT_ID,e.SAL
FROM EMPLOYEE e inner join DEPARTMENT d ON
e.DEPT_ID=d.DEPT_ID;

SELECT e.EMP_NAME,e.EMP_ID,d.DEPT_NAME,d.DEPT_ID,e.SAL
FROM EMPLOYEE e full outer join DEPARTMENT d ON
e.DEPT_ID=d.DEPT_ID;
```

```
SELECT * FROM EMPLOYEE WHERE EMP_ID>3
MINUS
SELECT * FROM EMPLOYEE WHERE EMP_ID>5

SELECT * FROM EMPLOYEE WHERE EMP_ID>5

/*see Departments names , Employee name  and maximum salary in each department */
SELECT e.EMP_NAME,d.DEPT_NAME,max(e.SAL)  AS MAX_SAL
FROM EMPLOYEE e INNER JOIN  DEPARTMENT d ON
e.DEPT_ID=d.DEPT_ID
GROUP BY(d.DEPT_NAME,e.EMP_NAME)
HAVING  max(e.SAL) > 70;

/*Display all department names and their eployee count*/
SELECT d.DEPT_NAME ,COUNT(e.EMP_ID) AS EMPLOYEE_COUNT
FROM EMPLOYEE e inner join DEPARTMENT d ON
e.DEPT_ID=d.DEPT_ID
GROUP BY (d.DEPT_NAME);

/*print out the names of the manager of each employee right beside the employee, we can use self join. */
SELECT E.EMP_NAME  AS EMPLOYEE_NAME,M.EMP_NAME  AS MANAGER_NAME
FROM EMPLOYEE E right outer JOIN EMPLOYEE M ON
E.EMP_ID=M.MGR_ID

/*How to generate row number in SQL Without ROWNUM*/
SELECT EMP_NAME,SAL,  (SELECT COUNT(*)  FROM EMPLOYEE i WHERE o.EMP_NAME >= i.EMP_NAME)
row_num
FROM EMPLOYEE o
order by row_num

/*Here the ROWNUM is ordered by emp_id*/
SELECT EMP_NAME,SAL,  ROWNUM
FROM EMPLOYEE

/*Display first 5 records in a table*/
SELECT *
FROM EMPLOYEE
WHERE ROWNUM <= 5;

/*Display first 5 records in a table witout using psudeo column ROWNUM*/
SELECT  emp_name
FROM EMPLOYEE o
WHERE (SELECT COUNT(*)  FROM EMPLOYEE i WHERE o.EMP_NAME >= i.EMP_NAME)<5

/*Suppose if you want to generate the row numbers in the order of ascending employee salaries for example, ROWNUM
will not work. But you may use ROW_NUMBER() OVER() like shown below:*/
SELECT EMP_name, sal, row_number() over(order by sal desc) rownum_by_sal FROM EMPLOYEE o

/*RANK does not assign unique numbers?nor does it assign contiguous numbers. If two records tie for second place, no
record will be assigned the 3rd rank as no one came in third, according to RANK.*/
SELECT EMP_name, sal, RANK() over(order by sal desc) rownum_by_sal FROM EMPLOYEE o;

/*DENSE_RANK,  like RANK, does not assign unique numbers, but it does assign contiguous numbers. Even though two
records tied for second place, there is a third-place record. */
SELECT EMP_name, sal, DENSE_RANK() over(order by sal desc) rownum_by_sal FROM EMPLOYEE o;
```

```
/*Display manager id , Manager name , and count of employees under each manager */
SELECT distinct m.MGR_ID,e.EMP_NAME  as mgr_name, COUNT(e.EMP_ID) OVER (PARTITION BY e.EMP_NAME) as
EMP_COUNT  FROM EMPLOYEE m INNER  JOIN EMPLOYEE e ON m.MGR_ID=e.EMP_ID

/*Select Values within a range derived from other Table*/

CREATE VIEW v1 AS SELECT MIN(col1) as min_val, MAX(col1) as max_val FROM table1;

SELECT t2.* FROM table2 t2, table1 t1 WHERE (t2.col1 >= t1.min_val AND t2.col1 <= t1.max_val);
```