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);