DBMS & SQL – GRADED ASSIGNMENT

1. **Create a table "student" with the structure/dictionary given above and insert 10 records given in the table created.**

   1. 1 Create a table "faculty" with the structure/dictionary given above and insert 8 records given in the table created.

   1.2 Create a table "course" with the structure/dictionary given above and insert 8 records given in the table created.

   1.3 Create a table "registration" with the structure/dictionary given above and insert 18 records given in the table created.

SQL QUERY:

STUDENT:

```
CREATE TABLE STUDENT (
SID VARCHAR(3) NOT NULL,
SNAME VARCHAR(10) NOT NULL,
SEX VARCHAR(3),
MAJOR VARCHAR(3),
GPA NUMBER(3,2)
);
==================================================================
-- STUDENT

INSERT INTO STUDENT VALUES (987, 'POIRIER', 'F', 'MGT', 3.2);
INSERT INTO STUDENT VALUES (763, 'PARKER', 'F', 'FIN', 2.7);
INSERT INTO STUDENT VALUES (218, 'RICHARDS', 'M', 'ACC', 2.4);
INSERT INTO STUDENT VALUES (359, 'PELNICK', 'F', 'FIN', 3.6);
INSERT INTO STUDENT VALUES (862, 'FAGIN', 'M', 'MGT', 2.2);
INSERT INTO STUDENT VALUES (748, 'MEGLIN', 'M', 'MGT', 2.8);
INSERT INTO STUDENT VALUES (506, 'LEE', 'M', 'FIN', 2.7);
INSERT INTO STUDENT VALUES (581, 'GAMBREL', 'F', 'MKT', 3.8);
INSERT INTO STUDENT VALUES (372, 'QUICK', 'F', 'MGT', 3.5);
INSERT INTO STUDENT VALUES (126, 'ANDERSON', 'M', 'ACC', 3.7);
```

Output:

	SID	SNAME	SEX	MAJOR	GPA
1	987	POIRIER	F	MGT	3.2
2	763	PARKER	F	FIN	2.7
3	218	RICHARDS	M	ACC	2.4
4	359	PELNICK	F	FIN	3.6
5	862	FAGIN	M	MGT	2.2
6	748	MEGLIN	M	MGT	2.8
7	506	LEE	M	FIN	2.7
8	581	GAMBREL	F	MKT	3.8
9	372	QUICK	F	MGT	3.5
10	126	ANDERSON	M	ACC	3.7

FACULTY:

```
CREATE TABLE FACULTY (
FID VARCHAR(3) NOT NULL,
FNAME VARCHAR(10) NOT NULL,
EXT VARCHAR(3),
DEPT VARCHAR(3),
RANK1 VARCHAR(4),
SALARY INTEGER
);
```

```
INSERT INTO FACULTY VALUES (036, 'BARGES', 325, 'MGT', 'ASSO', 35000);
INSERT INTO FACULTY VALUES (117, 'JARDIN', 212, 'FIN', 'FULL', 33000);
INSERT INTO FACULTY VALUES (098, 'KENEDY', 176, 'ACC', 'ASSO', 30000);
INSERT INTO FACULTY VALUES (075, 'SAMPLE', 171, 'MKT', 'ASST', 25000);
INSERT INTO FACULTY VALUES (138, 'WARD', 125, 'MGT', 'INST', 20000);
INSERT INTO FACULTY VALUES (219, 'PETERS', 220, 'FIN', 'FULL', 45000);
INSERT INTO FACULTY VALUES (151, 'DARDEN', 250, 'ACC', 'ASSO', 37000);
INSERT INTO FACULTY VALUES (113, 'PIERCE', 205, 'MGT', 'INST', 22000);
```

Output:

SQL | All Rows Fetched: 8 in 0.002 seconds

	FID	FNAME	EXT	DEPT	RANK1	SALARY
1	36	BARGES	325	MGT	ASSO	35000
2	117	JARDIN	212	FIN	FULL	33000
3	98	KENEDY	176	ACC	ASSO	30000
4	75	SAMPLE	171	MKT	ASST	25000
5	138	WARD	125	MGT	INST	20000
6	219	PETERS	220	FIN	FULL	45000
7	151	DARDEN	250	ACC	ASSO	37000
8	113	PIERCE	205	MGT	INST	22000

COURSE:

```
CREATE TABLE COURSE (
CRSNBR VARCHAR(6) NOT NULL,
CNAME VARCHAR(25) NOT NULL,
CREDIT INTEGER(1),
MAXENRL INTEGER,
FID VARCHAR(3) NOT NULL
);
```

```
INSERT INTO COURSE VALUES ('MGT630', 'INTRODUCTION TO MGMT', 4, 30, 138);
INSERT INTO COURSE VALUES ('FIN601', 'MANAGERIAL FINANCE', 4, 25, 117);
INSERT INTO COURSE VALUES ('MKT610', 'MARKETING FOR MANAGERS', 3, 35,
075);
INSERT INTO COURSE VALUES ('MKT661', 'TAXATION', 3, 30, 098);
INSERT INTO COURSE VALUES ('FIN602', 'INVESTMENT SKILLS', 3, 25, 219);
INSERT INTO COURSE VALUES ('ACC601', 'BASIC ACCOUNTING', 4, 25, 098);
INSERT INTO COURSE VALUES ('MGT681', 'INTERL. MANAGEMENT', 3, 20, 036);
INSERT INTO COURSE VALUES ('MKT670', 'PRODUCT MARKETING', 3, 20, 075);
```

Output:

SQL | All Rows Fetched: 8 in 0.003 seconds

	CRSNBR	CNAME	CREDIT	MAXENRL	FID
1	MGT630	INTRODUCTION TO MGMT	4	30	138
2	FIN601	MANAGERIAL FINANCE	4	25	117
3	MKT610	MARKETING FOR MANAGERS	3	35	75
4	MKT661	TAXATION	3	30	98
5	FIN602	INVESTMENT SKILLS	3	25	219
6	ACC601	BASIC ACCOUNTING	4	25	98
7	MGT681	INTERL. MANAGEMENT	3	20	36
8	MKT670	PRODUCT MARKETING	3	20	75

REGISTRATION:

```
CREATE TABLE REGISTRATION (
CRSNBR VARCHAR(6) NOT NULL,
SID VARCHAR(3) NOT NULL,
GRADE VARCHAR(1)
);
```

```
INSERT INTO REGISTRATION VALUES ('MGT630', 987, 'A');
INSERT INTO REGISTRATION VALUES ('FIN602', 987, 'B');
INSERT INTO REGISTRATION VALUES ('MKT610', 987, 'A');
INSERT INTO REGISTRATION VALUES ('FIN601', 763, 'B');
INSERT INTO REGISTRATION VALUES ('FIN602', 763, 'B');
INSERT INTO REGISTRATION VALUES ('ACC610', 763, 'B');
INSERT INTO REGISTRATION VALUES ('ACC610', 218, 'A');
INSERT INTO REGISTRATION VALUES ('ACC661', 218, 'A');
INSERT INTO REGISTRATION VALUES ('MGT630', 218, 'C');
INSERT INTO REGISTRATION VALUES ('MGT630', 359, 'F');
INSERT INTO REGISTRATION VALUES ('MGT681', 359, 'B');
INSERT INTO REGISTRATION VALUES ('MKT610', 359, 'A');
INSERT INTO REGISTRATION VALUES ('MKT610', 359, 'A');
INSERT INTO REGISTRATION VALUES ('MKT670', 862, 'A');
INSERT INTO REGISTRATION VALUES ('ACC610', 862, 'B');
```

```
INSERT INTO REGISTRATION VALUES ('MGT630', 748, 'C');
INSERT INTO REGISTRATION VALUES ('MGT681', 748, 'B');
INSERT INTO REGISTRATION VALUES ('FIN601', 748, 'A');
```

Output:

SQL | All Rows Fetched: 18 in 0.003 seconds

	CRSNBR	SID	GRADE
1	MGT630	987	A
2	FIN602	987	B
3	MKT610	987	A
4	FIN601	763	B
5	FIN602	763	B
6	ACC610	763	B
7	ACC610	218	A
8	ACC661	218	A
9	MGT630	218	C
10	MGT630	359	F
11	MGT681	359	B
12	MKT610	359	A
13	MKT610	359	A
14	MKT670	862	A
15	ACC610	862	B
16	MGT630	748	C
17	MGT681	748	B
18	FIN601	748	A

2. **Retrieve the list of students in alphabetical order**

SQL Query: select * from student order by sname asc;

Output:

SQL | All Rows Fetched: 10 in 0.002 seconds

	SID	SNAME	SEX	MAJOR	GPA
1	126	ANDERSON	M	ACC	3.7
2	862	FAGIN	M	MGT	2.2
3	581	GAMBREL	F	MKT	3.8
4	506	LEE	M	FIN	2.7
5	748	MEGLIN	M	MGT	2.8
6	763	PARKER	F	FIN	2.7
7	359	PELNICK	F	FIN	3.6
8	987	POIRIER	F	MGT	3.2
9	372	QUICK	F	MGT	3.5
10	218	RICHARDS	M	ACC	2.4

3. **Display a list of female students with a GPA above 3.25.**

SQL Query: select * from student where sex = 'F' and GPA > 3.25;

Output:

SQL | All Rows Fetched: 3 in 0.024 seconds

	SID	SNAME	SEX	MAJOR	GPA
1	359	PELNICK	F	FIN	3.6
2	581	GAMBREL	F	MKT	3.8
3	372	QUICK	F	MGT	3.5

4. **Retrieve the names, majors, and GPA of all students who have a GPA above 3.5 and who are majoring in either accounting or finance**

SQL Query: select sname, major, gpa from student where gpa >3.5 and (major = 'ACC' or major = 'FIN');

Output:

SQL | All Rows Fetched: 2 in 0.002 seconds

	SNAME	MAJOR	GPA
1	PELNICK	FIN	3.6
2	ANDERSON	ACC	3.7

5. **Next year every faculty member will receive a 5% salary increase. List the names of each faculty member, his/her current salary, and next years salary.**

SQL Query: select fname, salary as "CURRENT SALARY", salary * 1.05 as "NEXT YEAR SALARY" from faculty;

Output:

SQL | All Rows Fetched: 8 in 0.033 seconds

	FNAME	CURRENT SALARY	NEXT YEAR SALARY
1	BARGES	35000	36750
2	JARDIN	33000	34650
3	KENEDY	30000	31500
4	SAMPLE	25000	26250
5	WARD	20000	21000
6	PETERS	45000	47250
7	DARDEN	37000	38850
8	PIERCE	22000	23100

6. **Retrieve the average GPA from student where major='MGT'.**

SQL Query: select major, AVG(GPA)as "Average GPA" from student where major = 'MGT' group by major;

Output:

SQL | All Rows Fetched: 1 in 0.003 seconds

	MAJOR	Average GPA
1	MGT	2.925

7. **Create a new table rgn_copy and copy the data from the REGISTRATION table to the rgn_copy table. Change the grade to F in rgn_copy table where course no is MGT681.**

SQL Query:

```
create table rgn_copy as select * from registration;
update rgn_copy set grade = 'F' where crsnbr = 'MGT681';
select * from rgn_copy;
```

Output:

SQL | All Rows Fetched: 18 in 0.001 seconds

	CRSNBR	SID	GRADE
1	MGT630	987	A
2	FIN602	987	B
3	MKT610	987	A
4	FIN601	763	B
5	FIN602	763	B
6	ACC610	763	B
7	ACC610	218	A
8	ACC661	218	A
9	MGT630	218	C
10	MGT630	359	F
11	MGT681	359	F
12	MKT610	359	A
13	MKT610	359	A
14	MKT670	862	A
15	ACC610	862	B
16	MGT630	748	C
17	MGT681	748	F
18	FIN601	748	A

8. **Create a new table std_copy and copy the data from the student table to the std_copy table. A student whose ID number is 748 leaves the University. First delete the course in which student 748 is enrolled from the rgn_copy table. Then remove the student from the table std_copy.**

SQL Query:

```
create table std_copy as select * from student;

--delete course of student 748 from rgn_copy

delete from rgn_copy where sid = 748;
select * from rgn_copy;
```

Output:

SQL | All Rows Fetched: 15 in 0.002 seconds

	CRSNBR	SID	GRADE
1	MGT630	987	A
2	FIN602	987	B
3	MKT610	987	A
4	FIN601	763	B
5	FIN602	763	B
6	ACC610	763	B
7	ACC610	218	A
8	ACC661	218	A
9	MGT630	218	C
10	MGT630	359	F
11	MGT681	359	F
12	MKT610	359	A
13	MKT610	359	A
14	MKT670	862	A
15	ACC610	862	B

```
-- remove student 748 from the std_copy table

delete from std_copy where sid = 748;
select * from std_copy;
```

Output:

SQL | All Rows Fetched: 9 in 0.002 seconds

	SID	SNAME	SEX	MAJOR	GPA
1	987	POIRIER	F	MGT	3.2
2	763	PARKER	F	FIN	2.7
3	218	RICHARDS	M	ACC	2.4
4	359	PELNICK	F	FIN	3.6
5	862	FAGIN	M	MGT	2.2
6	506	LEE	M	FIN	2.7
7	581	GAMBREL	F	MKT	3.8
8	372	QUICK	F	MGT	3.5
9	126	ANDERSON	M	ACC	3.7

9. **Delete the tables rgn_copy and std_copy from the database**

SQL Query: drop table rgn_copy;
SQL Query: drop table std_copy;

Output:

```
Table RGN_COPY dropped.


Table STD_COPY dropped.
```

10. **Create a table IPMFAC with the following structure:**
FID Varchar2(3) where null values are not allowed; FNAME Varchar2(10) where null values are not allowed, EXT Varchar2(3) where null values are not allowed, DEPT Varchar2(3), RANK1 Varchar2(4), SALARY as integer. In this table, FID is the primary key.

SQL Query:

```
CREATE TABLE IPMFAC (
  FID VARCHAR2(3) NOT NULL,
  FNAME VARCHAR2(10) NOT NULL,
  EXT VARCHAR2(3) NOT NULL,
  DEPT VARCHAR2(3),
  RANK1 VARCHAR2(4),
  SALARY INTEGER,
  CONSTRAINT PK_IPMFAC PRIMARY KEY (FID)
);
```

Output:

```
Table IPMFAC created.
```

11. **Create a table IPMCO with the following structure:**
    CRSNBR Varchar2(6) with null values not allowed, CNAME Varchar2 25) with null values not allowed, CREDIT as integer, MAXENRL as integer, FID Varchar2(3) with null values not allowed. Now, introduce FID as Foreign Key and then reference to IPMFAC table considering FID of IPMFAC table and FID of IPMCO as common field.

    SQL Query:

```
CREATE TABLE IPMCO (
  CRSNBR VARCHAR2(6) NOT NULL,
  CNAME VARCHAR2(25) NOT NULL,
  CREDIT INTEGER,
  MAXENRL INTEGER,
  FID VARCHAR2(3) NOT NULL,
  CONSTRAINT PK_IPMCO PRIMARY KEY (CRSNBR),
  CONSTRAINT FK_IPMCO_IPMFAC FOREIGN KEY (FID) REFERENCES IPMFAC(FID)
);
```

Output:

```
Table IPMCO created.
```

12. **Create a view "Roster" that enables the individual to visualize selected data from the STUDENT, REGISTRATION, COURSE and FACULTY tables as being one table, This view includes course number, course name, name of person teaching the course, student ID and student name. Display course number, course name, student ID, and student name from view "Roster" for the course number "FIN601"**

    SQL Query:

```
CREATE VIEW Roster AS
SELECT c.CRSNBR, c.CNAME, f.FNAME AS "FACULTY NAME", s.SID,
s.SNAME
FROM COURSE c
JOIN REGISTRATION r ON c.CRSNBR = r.CRSNBR
JOIN STUDENT s ON r.SID = s.SID
JOIN FACULTY f ON c.FID = f.FID;
```

Output:

```
View ROSTER created.
```

SQL Query:

```
SELECT CRSNBR, CNAME, SID, SNAME
FROM Roster
WHERE CRSNBR = 'FIN601';
```

Output:

Script Output × | Query Result ×

SQL | All Rows Fetched: 2 in 0.009 seconds

	CRSNBR	CNAME	SID	SNAME
1	FIN601	MANAGERIAL FINANCE	763	PARKER
2	FIN601	MANAGERIAL FINANCE	748	MEGLIN

13. **Create an index "MAJORIND" using the MAJOR column of Student to improve performance, MAJOR descending.**

SQL Query:

```
create index MAJORIND on student (major desc);
```

Output:

```
Index MAJORIND created.
```

14. **Write a stored procedure named "Getstudents" : To list all the sname of table Student**

SQL Query:

```
CREATE OR REPLACE PROCEDURE Getstudents
AS
BEGIN
  for i in (select sname from student)
  loop
    dbms_output.put_line(i.sname);
  end loop;
END;
/
exec Getstudents;
```

Output:

```
Procedure GETSTUDENTS compiled


PL/SQL procedure successfully completed.
```

```
Connecting to the database DSML.
POIRIER
PARKER
RICHARDS
PELNICK
FAGIN
MEGLIN
LEE
GAMBREL
QUICK
ANDERSON
Process exited.
Disconnecting from the database DSML.
```

15. Create trigger, "salary_changes" to display the following information:

Old salary:
New salary:
Salary difference:

The trigger will be fired when the salary difference is observed in the faculty table.

SQL Query:

```
CREATE OR REPLACE TRIGGER salary_changes
BEFORE UPDATE OF salary ON Faculty
FOR EACH ROW
DECLARE
  old_salary NUMBER;
  new_salary NUMBER;
  salary_difference NUMBER;
BEGIN
  old_salary := :OLD.salary;
  new_salary := :NEW.salary;
  salary_difference := new_salary - old_salary;

  IF salary_difference <> 0 THEN
    DBMS_OUTPUT.PUT_LINE('Old salary: ' || old_salary);
    DBMS_OUTPUT.PUT_LINE('New salary: ' || new_salary);
    DBMS_OUTPUT.PUT_LINE('Salary difference: ' || salary_difference);
  END IF;
END;
/
```

--**update query**

update faculty set salary = salary + 1000 where FID = 75;

Output:

```
Trigger SALARY_CHANGES compiled

Old salary: 25000
New salary: 26000
Salary difference: 1000


1 row updated.
```