Assignment- SQL Concepts & Fundamentals

Solving Q1. To Q4. For All Tables

- CREATE TABLE
- INSERT INTO TABLE
- DISPLAY TABLE

1. CREATE TABLE StudentBasicInformation

```
CREATE TABLE StudentBasicInformation
(
   StudentName        VARCHAR(20)    NOT NULL,
   StudentSurName      VARCHAR(20),
   StudentRollNo       INT,
   StudentAddress      VARCHAR(30),
   StudentBdate        DATE,
   StudentAge          INT,
   StudentSex          CHAR(1),
   PRIMARY KEY(StudentRollNo)
);
```

SQL Worksheet

```
-- CREATE DATABASE STUDENT;     NOT AVAILABLE IN ORACLE LIVE SQL

CREATE TABLE StudentBasicInformation
(
    StudentName         VARCHAR(20)     NOT NULL,
    StudentSurName      VARCHAR(20),
    StudentRollNo       INT,
    StudentAddress      VARCHAR(30),
    StudentBdate        DATE,
    StudentMobile       NUMBER(10)      NOT NULL,
    StudentSex          CHAR(1),
    PRIMARY KEY(StudentRollNo)
)
```

```
Table created.
```

2. INSERT DATA INTO StudentBasicInformation

SQL Worksheet

Clear Find

```sql
VALUES (5, 'Nishant', 'Chauhan', 'F', TO_DATE('01/10/1993', 'DD/MM/YYYY'), 9912819116, 'Flat-11/4 Ramya Building,Mumbai'


INSERT INTO StudentBasicInformation
(StudentRollNo, StudentName, StudentSurName, StudentSex, StudentBdate, StudentMobile, StudentAddress)
VALUES (6, 'Priyanka', 'Chopra', 'F', TO_DATE('16/03/1998', 'DD/MM/YYYY'), 9910019116, 'Flat-101 Santok Tower,Mumbai');


INSERT INTO StudentBasicInformation
(StudentRollNo, StudentName, StudentSurName, StudentSex, StudentBdate, StudentMobile, StudentAddress)
VALUES (7, 'Virat', 'Kohli', 'F', TO_DATE('12/05/1993', 'DD/MM/YYYY'), 9912800116, 'Flat-11/4 Paschim Vihar,Delhi');


INSERT INTO StudentBasicInformation
(StudentRollNo, StudentName, StudentSurName, StudentSex, StudentBdate, StudentMobile, StudentAddress)
VALUES (8, 'M.S', 'Dhoni', 'M', TO_DATE('07/08/1991', 'DD/MM/YYYY'), 9912800777, 'House-119 Sarita Vihar,Ranchi');


INSERT INTO StudentBasicInformation
(StudentRollNo, StudentName, StudentSurName, StudentSex, StudentBdate, StudentMobile, StudentAddress)
VALUES (9, 'Alia', 'Bhatt', 'F', TO_DATE('11/05/1997', 'DD/MM/YYYY'), 9900800116, 'Flat-23/9 Paschim Vihar,Delhi');


INSERT INTO StudentBasicInformation
(StudentRollNo, StudentName, StudentSurName, StudentSex, StudentBdate, StudentMobile, StudentAddress)
VALUES (10, 'Saadhvi', 'Mehra', 'F', TO_DATE('11/05/1997', 'DD/MM/YYYY'), 9900822116, 'House-192 Janak Puri,Delhi');
```

1 row(s) inserted.

3. SHOW StudentBasicInformation

SELECT * FROM StudentBasicInformation;

```sql
SELECT * FROM StudentBasicInformation;
```

STUDENTNAME	STUDENTSURNAME	STUDENTROLLNO	STUDENTADDRESS	STUDENTBDATE	STUDENTMOBILE	STUDENTSEX
Ram	Gopal	1	123/5 Bhopal,MP	12-JAN-16	9812347816	M
Ramya	Goel	2	House-995 Ashok Vihar,Delhi	02-FEB-00	8818847816	F
Sanya	Agarwal	3	House-11 Preet Vihar,Gujarat	01-FEB-94	7812847816	F
Deepak	Chauhan	4	812/11 Ram Nagar,Gujarat	11-AUG-95	7812819116	M
Priyanka	Chopra	6	Flat-101 Santok Tower,Mumbai	16-MAR-98	9910019116	F
Virat	Kohli	7	Flat-11/4 Paschim Vihar,Delhi	12-MAY-93	9912800116	F
M.S	Dhoni	8	House-119 Sarita Vihar,Ranchi	07-AUG-91	9912800777	M
Alia	Bhatt	9	Flat-23/9 Paschim Vihar,Delhi	11-MAY-97	9900800116	F
Saadhvi	Mehra	10	House-192 Janak Puri,Delhi	11-MAY-97	9900822116	F

Download CSV

9 rows selected.

4. CREATE TABLE StudentAdmissionPaymentDetails

```
CREATE TABLE StudentAdmissionPaymentDetails
(
StudentRollNo   INT                 NOT NULL,
AmountPaid              DECIMAL(8,2),
AmountBalance           DECIMAL(8,2),
TransactionID           NUMBER(6),
PaymentMode             VARCHAR(7),
TransactionTime         TIMESTAMP,
Semester                INT,
PRIMARY KEY (TransactionID),
FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)
)
```

SQL Worksheet

```
-- SELECT * FROM StudentBasicInformation;

CREATE TABLE StudentAdmissionPaymentDetails
(
StudentRollNo   INT     NOT NULL,
AmountPaid      DECIMAL(8,2),
AmountBalance   DECIMAL(8,2),
TransactionID   NUMBER(6),
PaymentMode     VARCHAR(7),
TransactionTime TIMESTAMP,
Semester        INT,
PRIMARY KEY (TransactionID),
FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)
)
```

```
Table created.
```

5. INSERT DATA INTO StudentBasicInformation

INSERT INTO StudentAdmissionPaymentDetails
(TransactionID, StudentRollNo, AmountPaid, AmountBalance, TransactionTime, PaymentMode, Semester)
VALUES (101001, 1, 10000.00, 0.00, TIMESTAMP '2019-01-31 09:26:50.12' , 'Offline', 2)

```sql
INSERT INTO StudentAdmissionPaymentDetails
(TransactionID, StudentRollNo, AmountPaid, AmountBalance, TransactionTime, PaymentMode, Semester)
VALUES (101005, 2, 10030.00, 0.00, TIMESTAMP '2019-08-21 15:20:22.32' , 'Online', 3)

INSERT INTO StudentAdmissionPaymentDetails
(TransactionID, StudentRollNo, AmountPaid, AmountBalance, TransactionTime, PaymentMode, Semester)
VALUES (101006, 3, 10030.00, 20.00, TIMESTAMP '2019-09-01 05:01:02.12' , 'Offline', 3)

INSERT INTO StudentAdmissionPaymentDetails
(TransactionID, StudentRollNo, AmountPaid, AmountBalance, TransactionTime, PaymentMode, Semester)
VALUES (101007, 4, 10030.00, 50.00, TIMESTAMP '2018-08-01 06:01:32.52' , 'Online', 1)

INSERT INTO StudentAdmissionPaymentDetails
(TransactionID, StudentRollNo, AmountPaid, AmountBalance, TransactionTime, PaymentMode, Semester)
VALUES (101008, 4, 10000.00, 50.00, TIMESTAMP '2019-07-29 14:01:33.42' , 'Offline', 3)

INSERT INTO StudentAdmissionPaymentDetails
(TransactionID, StudentRollNo, AmountPaid, AmountBalance, TransactionTime, PaymentMode, Semester)
```

```
1 row(s) inserted.
```

6. SHOW StudentBasicInformation

SELECT * FROM StudentBasicInformation;

```sql
SELECT * FROM StudentAdmissionPaymentDetails order by transactionid ;
```

STUDENTROLLNO	AMOUNTPAID	AMOUNTBALANCE	TRANSACTIONID	PAYMENTMODE	TRANSACTIONTIME	SEMESTER
1	10000	0	101001	Offline	31-JAN-19 09.26.50.120000 AM	2
1	9000	1000	101002	Online	31-JAN-20 11.26.50.120000 AM	4
2	10030	0	101003	Online	31-MAR-18 03.26.22.320000 PM	2
3	10030	20	101004	Offline	21-FEB-18 05.16.02.120000 AM	2
2	10030	0	101005	Online	21-AUG-19 03.20.22.320000 PM	3
3	10030	20	101006	Offline	01-SEP-19 05.01.02.120000 AM	3
4	10030	50	101007	Online	01-AUG-18 06.01.32.520000 AM	1
4	10000	50	101008	Offline	29-JUL-19 02.01.33.420000 PM	3
5	10050	20	101009	Offline	01-SEP-18 03.01.02.110000 PM	5
6	10010	40	101010	Offline	01-SEP-20 09.59.02.120000 AM	5

Download CSV

7. CREATE TABLE StudentSubjectInformation

```
CREATE TABLE StudentSubjectInformation
(
SubjectOpted                VARCHAR(10),
StudentRollNo               INT,
SubjectTotalMarks           INT,
SubjectObtainedMarks        INT,
StudentMarksPercentage      NUMERIC(4,2),
Semester                    INT,
PRIMARY KEY (SubjectOpted,StudentRollNo),
FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)
)
```

```
CREATE TABLE StudentSubjectInformation
(
SubjectOpted            VARCHAR(10),
StudentRollNo           INT,
SubjectTotalMarks       INT,
SubjectObtainedMarks    INT,
StudentMarksPercentage  NUMERIC(4,2),
Semester                INT,
PRIMARY KEY (SubjectOpted,StudentRollNo),
FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)
)
```

```
Table created.
```

8. INSERT ALL StudentSubjectInformation

```
INSERT ALL
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DBMS', 3, 89, 100, 1)
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DS', 3, 55, 70, 2 )
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DBMS', 4, 90, 100, 1)
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('ALGO', 4, 60, 70, 3 )
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('ALGO', 5, 86, 100, 3)
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DS', 5, 62, 70, 2 )
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DBMS', 1, 91, 100, 1)
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DS', 2, 69, 80, 2 )
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('DBMS', 2, 50, 70, 1)
INTO StudentSubjectInformation
(SubjectOpted, StudentRollNo, SubjectObtainedMarks, SubjectTotalMarks, Semester) VALUES  ('ALGO', 1, 60, 70, 3 )
SELECT * FROM dual;
```

10 row(s) inserted.

9. SHOW StudentSubjectInformation

SELECT * FROM StudentSubjectInformation;

31

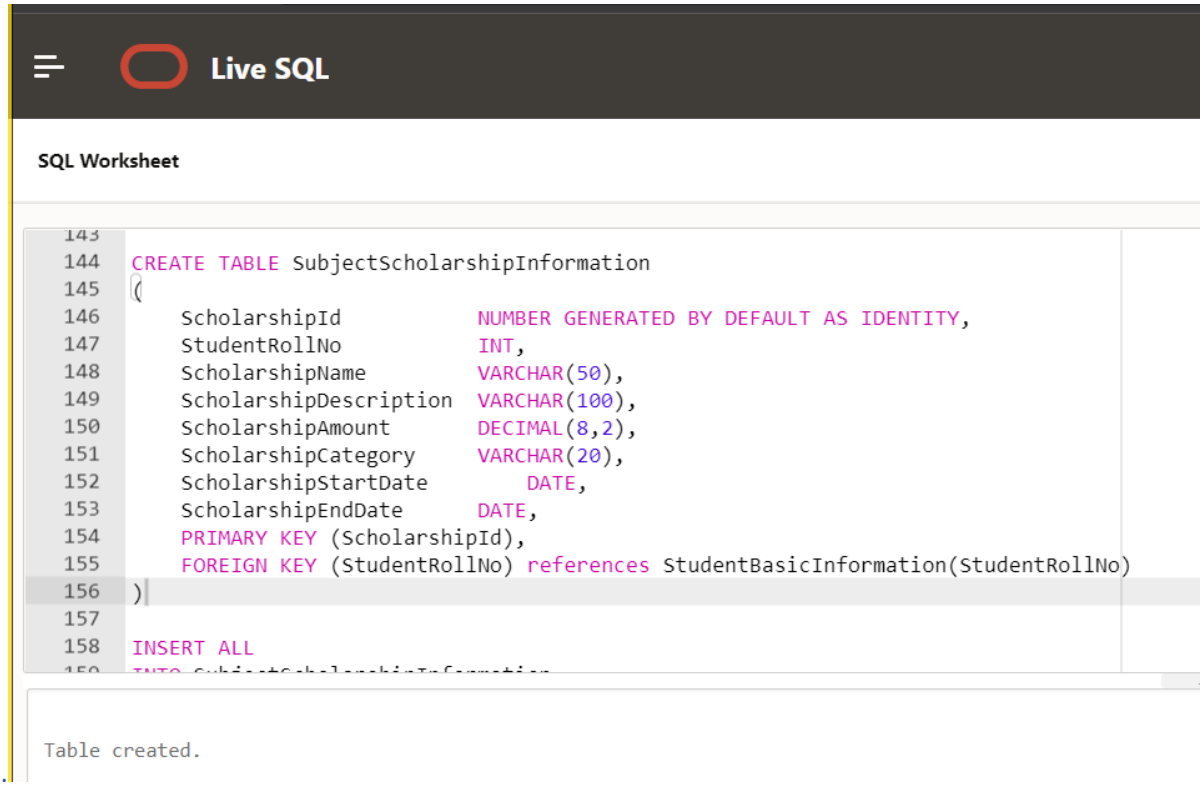
SUBJECTOPTED	STUDENTROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	3	100	89	-	1
DS	3	70	55	-	2
DBMS	4	100	90	-	1
ALGO	4	70	60	-	3
ALGO	5	100	86	-	3
DS	5	70	62	-	2
DBMS	1	100	91	-	1
DS	2	80	69	-	2
DBMS	2	70	50	-	1
ALGO	1	70	60	-	3

Download CSV

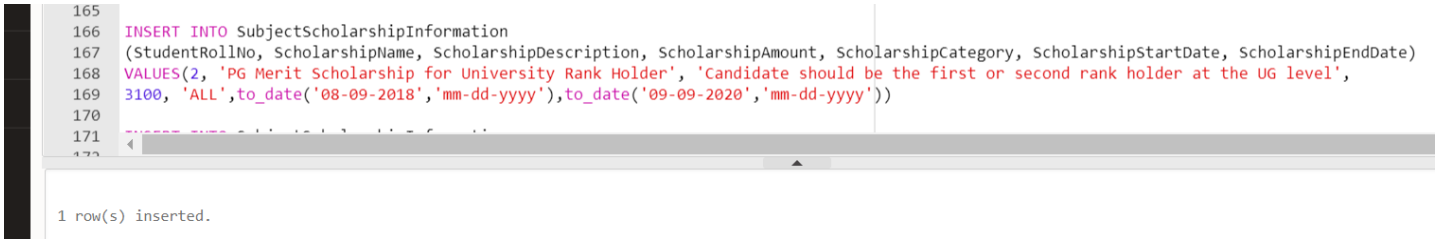
10 rows selected.

10. CREATE TABLE SubjectScholarshipInformation

```
CREATE TABLE SubjectScholarshipInformation
(
   ScholarshipId            NUMBER GENERATED BY DEFAULT AS IDENTITY,
   StudentRollNo                    INT,
   ScholarshipName                  VARCHAR(70),
   ScholarshipDescription           VARCHAR(100),
   ScholarshipAmount                DECIMAL(8,2),
   ScholarshipCategory              VARCHAR(20),
   ScholarshipStartDate             DATE,
   ScholarshipEndDate               DATE,
   PRIMARY KEY (ScholarshipId),
   FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)
)
```



11. INSERT INTO SubjectScholarshipInformation



12. SHOW SubjectScholarshipInformation

SELECT * FROM SubjectScholarshipInformation;

SCHOLARSHIPID	STUDENTROLLNO	SCHOLARSHIPNAME	SCHOLARSHIPDESCRIPTION	SCHOLARSHIPAMOUNT	SCHOLARSHIPCATEGORY	SCHOLARSHIPSTARTDATE	SCHOLARSHIPENDDATE
4	3	Indira Gandhi Scholarship Scheme	The scheme is applicable to such a single girl child who has taken admission in regular	36200	ALL	11-JUN-18	11-JUN-19
5	5	PG Merit Scholarship for University Rank Holder	Candidate should be the first or second rank holder at the UG level	3100	ALL	09-FEB-17	09-FEB-19
6	6	Emeritus Fellowship	The candidate must have given their service career with quality research and distributed work.	8000	SC	01-SEP-19	02-AUG-20
7	8	Post Graduate Merit Scholarship for University Rank Holder	Candidate should be the first or second rank holder at the UG level	3100	ALL	09-OCT-18	09-OCT-20
8	7	Emeritus Fellowship	The candidate must have given their service career with quality research and distributed work.	8000	SC	01-JAN-20	01-JAN-21
11	1	CV Raman Scholarship Scheme	The scheme is for a child who has taken admission in regular, full-time PG	5200	HANDICAPPED	11-AUG-17	11-AUG-18
1	1	Indira Gandhi Scholarship Scheme	The scheme is for a single girl child who has taken admission in regular, full-time	36200	ALL	11-AUG-19	11-AUG-20
10	10	Dr.D.S.Kothari Postdoctoral Fellowship Scheme	The fellowship is for applicants <35 yrs age who has Ph.D. degree in science faculty.	2000	SC	19-MAY-20	19-MAY-21

↕

SCID	ROLLN	SCNAME	DESCRIPTION	SCAMOUNT	CATEGORY	STARTDATE	ENDDATE
4	3	Indira Gandhi Scholars	The scheme is ap	36200	ALL	11-JUN-18	11-JUN-19
5	5	PG Merit Scholarship f	Candidate should	3100	ALL	09-FEB-17	09-FEB-19
6	6	Emeritus Fellowship	The candidate mu	8000	SC	01-SEP-19	02-AUG-20
7	8	Post Graduate Merit S	Candidate should	3100	ALL	09-OCT-18	09-OCT-20
8	7	Emeritus Fellowship	The candidate mu	8000	SC	01-JAN-20	01-JAN-21
11	1	CV Raman Scholarship	The scheme is for	5200	HANDICAPPED	11-AUG-17	11-AUG-18
1	1	Indira Gandhi Scholars	The scheme is for	36200	ALL	11-AUG-19	11-AUG-20
10	10	Dr.D.S.Kothari Postdo	The fellowship is f	2000	SC	19-MAY-20	19-MAY-21
3	4	Special Scholarship Sc	The annual income	7200	OBC	01-SEP-17	01-AUG-18
9	9	Post-Doctoral Fellows	The fellowship is f	5000	ST	11-JAN-19	11-JAN-21

Solving Q5 & Q6: Updating

I. MODIFYING Table *StudentBasicInformation*

1. UPDATE StudentBasicInformation SET STUDENTSEX = 'M' WHERE STUDENTNAME = 'Nishant' OR STUDENTNAME = 'Virat';
2. UPDATE StudentBasicInformation SET STUDENTBDATE = TO_DATE('01/02/2001', 'DD/MM/YYYY') WHERE STUDENTROLLNO = 2;

- BEFORE

```
SELECT * FROM StudentBasicInformation;
```

STUDENTNAME	STUDENTSURNAME	STUDENTROLLNO	STUDENTADDRESS	STUDENTBDATE	STUDENTMOBILE	STUDENTSEX
Ram	Gopal	1	123/5 Bhopal,MP	12-JAN-16	9812347816	M
Ramya	Goel	2	House-995 Ashok Vihar,Delhi	02-FEB-00	8818847816	F
Sanya	Agarwal	3	House-11 Preet Vihar,Gujarat	01-FEB-94	7812847816	F
Deepak	Chauhan	4	812/11 Ram Nagar,Gujarat	11-AUG-95	7812819116	M
Nishant	Chauhan	5	11/4 Ramya Building,Mumbai	01-OCT-93	9912819116	F
Priyanka	Chopra	6	Flat-101 Santok Tower,Mumbai	16-MAR-98	9910019116	F
Virat	Kohli	7	Flat-11/4 Paschim Vihar,Delhi	12-MAY-93	9912800116	F
M.S	Dhoni	8	House-119 Sarita Vihar,Ranchi	07-AUG-91	9912800777	M
Alia	Bhatt	9	Flat-23/9 Paschim Vihar,Delhi	11-MAY-97	9900800116	F
Saadhvi	Mehra	10	House-192 Janak Puri,Delhi	11-MAY-97	9900822116	F

Download CSV
10 rows selected.

- AFTER

```
SELECT * FROM StudentBasicInformation;

-- UPDATE StudentBasicInformation SET STUDENTSEX = 'M' WHERE STUDENTNAME = 'Nishant' OR STUDENTNAME = 'Virat';
-- UPDATE StudentBasicInformation SET STUDENTBDATE = TO_DATE('01/02/2001', 'DD/MM/YYYY') WHERE STUDENTROLLNO = 2;
```

STUDENTNAME	STUDENTSURNAME	STUDENTROLLNO	STUDENTADDRESS	STUDENTBDATE	STUDENTMOBILE	STUDENTSEX
Ram	Gopal	1	123/5 Bhopal,MP	12-JAN-16	9812347816	M
Ramya	Goel	2	House-995 Ashok Vihar,Delhi	01-FEB-01	8818847816	F
Sanya	Agarwal	3	House-11 Preet Vihar,Gujarat	01-FEB-94	7812847816	F
Deepak	Chauhan	4	812/11 Ram Nagar,Gujarat	11-AUG-95	7812819116	M
Nishant	Chauhan	5	11/4 Ramya Building,Mumbai	01-OCT-93	9912819116	M
Priyanka	Chopra	6	Flat-101 Santok Tower,Mumbai	16-MAR-98	9910019116	F
Virat	Kohli	7	Flat-11/4 Paschim Vihar,Delhi	12-MAY-93	9912800116	M
M.S	Dhoni	8	House-119 Sarita Vihar,Ranchi	07-AUG-91	9912800777	M
Alia	Bhatt	9	Flat-23/9 Paschim Vihar,Delhi	11-MAY-97	9900800116	F
Saadhvi	Mehra	10	House-192 Janak Puri,Delhi	11-MAY-97	9900822116	F

Download CSV
10 rows selected.

II. MODIFYING Table *StudentAdmissionPaymentDetails*

3. UPDATE StudentAdmissionPaymentDetails SET AMOUNTPAID = 12000
   WHERE StudentRollNo IN (3,5);
4. UPDATE StudentAdmissionPaymentDetails SET AMOUNTBALANCE = 12000-AMOUNTPAID;

- BEFORE

```
SELECT * FROM StudentAdmissionPaymentDetails order by transactionid ;
```

STUDENTROLLNO	AMOUNTPAID	AMOUNTBALANCE	TRANSACTIONID	PAYMENTMODE	TRANSACTIONTIME	SEMESTER
1	10000	0	101001	Offline	31-JAN-19 09.26.50.120000 AM	2
1	9000	1000	101002	Online	31-JAN-20 11.26.50.120000 AM	4
2	10030	0	101003	Online	31-MAR-18 03.26.22.320000 PM	2
3	10030	20	101004	Offline	21-FEB-18 05.16.02.120000 AM	2
2	10030	0	101005	Online	21-AUG-19 03.20.22.320000 PM	3
3	10030	20	101006	Offline	01-SEP-19 05.01.02.120000 AM	3
4	10030	50	101007	Online	01-AUG-18 06.01.32.520000 AM	1
4	10000	50	101008	Offline	29-JUL-19 02.01.33.420000 PM	3
5	10050	20	101009	Offline	01-SEP-18 03.01.02.110000 PM	5
6	10010	40	101010	Offline	01-SEP-20 09.59.02.120000 AM	5

Download CSV
10 rows selected.

- AFTER

```
UPDATE StudentAdmissionPaymentDetails SET AMOUNTPAID = 12000
WHERE StudentRollNo IN (3,5);

UPDATE StudentAdmissionPaymentDetails SET AMOUNTBALANCE = 12000-AMOUNTPAID;

SELECT * FROM StudentAdmissionPaymentDetails
```

STUDENTROLLNO	AMOUNTPAID	AMOUNTBALANCE	TRANSACTIONID	PAYMENTMODE	TRANSACTIONTIME	SEMESTER
1	10000	2000	101001	Offline	31-JAN-19 09.26.50.120000 AM	2
1	9000	3000	101002	Online	31-JAN-20 11.26.50.120000 AM	4
2	10030	1970	101003	Online	31-MAR-18 03.26.22.320000 PM	2
3	12000	0	101004	Offline	21-FEB-18 05.16.02.120000 AM	2
2	10030	1970	101005	Online	21-AUG-19 03.20.22.320000 PM	3
3	12000	0	101006	Offline	01-SEP-19 05.01.02.120000 AM	3
4	10030	1970	101007	Online	01-AUG-18 06.01.32.520000 AM	1
4	10000	2000	101008	Offline	29-JUL-19 02.01.33.420000 PM	3
5	12000	0	101009	Offline	01-SEP-18 03.01.02.110000 PM	5
6	10010	1990	101010	Offline	01-SEP-20 09.59.02.120000 AM	5

Download CSV
10 rows selected.

III. MODIFYING Table *StudentSubjectInformation*

5. UPDATE StudentSubjectInformation SET STUDENTMARKSPERCENTAGE = SUBJECTOBTAINEDMARKS/SUBJECTTOTALMARKS*100;

- BEFORE

```
SELECT * FROM StudentSubjectInformation;
```

SUBJECTOPTED	STUDENTROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	3	100	89	-	1
DS	3	70	55	-	2
DBMS	4	100	90	-	1
ALGO	4	70	60	-	3
ALGO	5	100	86	-	3
DS	5	70	62	-	2
DBMS	1	100	91	-	1
DS	2	80	69	-	2
DBMS	2	70	50	-	1
ALGO	1	70	60	-	3

Download CSV
10 rows selected.

- AFTER

```
SELECT * FROM StudentSubjectInformation;

UPDATE StudentSubjectInformation SET STUDENTMARKSPERCENTAGE = SUBJECTOBTAINEDMARKS/SUBJECTTOTALMARKS*100;
```

SUBJECTOPTED	STUDENTROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	3	100	89	89	1
DS	3	70	55	78.57	2
DBMS	4	100	90	90	1
ALGO	4	70	60	85.71	3
ALGO	5	100	86	86	3
DS	5	70	62	88.57	2
DBMS	1	100	91	91	1
DS	2	80	69	86.25	2
DBMS	2	70	50	71.43	1
ALGO	1	70	60	85.71	3

Download CSV
10 rows selected.

IV. MODIFYING Table *SubjectScholarshipInformation*

6. UPDATE SubjectScholarshipInformation
   SET SCHOLARSHIPAMOUNT=35000
   WHERE SCHOLARSHIPNAME = 'Indira Gandhi Scholarship Scheme';

- BEFORE

SQL Worksheet

Clear Find Actions Save Run

```
SELECT * FROM SubjectScholarshipInformation WHERE SCHOLARSHIPNAME='Indira Gandhi Scholarship Scheme';
```

Resize Code Editor

SCHOLARSHIPID	STUDENTROLLNO	SCHOLARSHIPNAME	SCHOLARSHIPDESCRIPTION	SCHOLARSHIPAMOUNT	SCHOLARSHIPCATEGORY	SCHOLARSHIPSTARTDATE	SCHOLARSHIPENDDATE
1	1	Indira Gandhi Scholarship Scheme	The scheme is for a single girl child who has taken admission in regular, full-time	36200	ALL	11-AUG-19	11-AUG-20
4	3	Indira Gandhi Scholarship Scheme	The scheme is applicable to such a single girl child who has taken admission in regular	36200	ALL	11-JUN-18	11-JUN-19

Download CSV
2 rows selected.

- AFTER

```
SELECT * FROM SubjectScholarshipInformation WHERE SCHOLARSHIPNAME='Indira Gandhi Scholarship Scheme';

UPDATE SubjectScholarshipInformation SET SCHOLARSHIPAMOUNT=35000 WHERE SCHOLARSHIPNAME='Indira Gandhi Scholarship Scheme';
```

SCHOLARSHIPID	STUDENTROLLNO	SCHOLARSHIPNAME	SCHOLARSHIPDESCRIPTION	SCHOLARSHIPAMOUNT	SCHOLARSHIPCATEGORY	SCHOLARSHIPSTARTDATE	SCHOLARSHIPENDDATE
1	1	Indira Gandhi Scholarship Scheme	The scheme is for a single girl child who has taken admission in regular, full-time	35000	ALL	11-AUG-19	11-AUG-20
4	3	Indira Gandhi Scholarship Scheme	The scheme is applicable to such a single girl child who has taken admission in regular	35000	ALL	11-JUN-18	11-JUN-19

Download CSV
2 rows selected.

Solving Q7:

- Query:

```
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo IN
(
   SELECT StudentRollNo FROM SubjectScholarshipInformation
   WHERE ScholarshipAmount > 5000
);
```

- Output:

```
-- Students with Scholarship Amount > 5000
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo IN
(
    SELECT StudentRollNo FROM SubjectScholarshipInformation
    WHERE ScholarshipAmount > 5000
);
```

STUDENTNAME	STUDENTSURNAME	STUDENTROLLNO	STUDENTADDRESS	STUDENTBDATE	STUDENTMOBILE	STUDENTSEX
Ram	Gopal	1	123/5 Bhopal,MP	12-JAN-16	9812347816	M
Sanya	Agarwal	3	House-11 Preet Vihar,Gujarat	01-FEB-94	7812847816	F
Priyanka	Chopra	6	Flat-101 Santok Tower,Mumbai	16-MAR-98	9910019116	F
Deepak	Chauhan	4	812/11 Ram Nagar,Gujarat	11-AUG-95	7812819116	M
Virat	Kohli	7	Flat-11/4 Paschim Vihar,Delhi	12-MAY-93	9912800116	M

Download CSV

5 rows selected.

Solving Q8:

- Query:

```
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo NOT IN
(
    SELECT StudentRollNo FROM SubjectScholarshipInformation
)
```

OR

```
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo NOT IN
(
    SELECT si.StudentRollNo
    FROM StudentBasicInformation si, SubjectScholarshipInformation ss
    WHERE si.StudentRollNo=ss.StudentRollNo
)
```

- Output:

```
-- Students not got the scholarship
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo NOT IN
(
    SELECT StudentRollNo FROM SubjectScholarshipInformation
)
```

STUDENTNAME	STUDENTSURNAME	STUDENTROLLNO	STUDENTADDRESS	STUDENTBDATE	STUDENTMOBILE	STUDENTSEX
Ramya	Goel	2	House-995 Ashok Vihar,Delhi	01-FEB-01	8818847816	F

Download CSV

Solving Q9:

PROCEDURE: TO FILL Percentage

- CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE updatePercentage
IS
BEGIN
   UPDATE StudentSubjectInformation SET
   StudentMarksPercentage = SubjectObtainedMarks/SubjectTotalMarks*100;
   DBMS_OUTPUT.PUT_LINE('PERCENTAGE UPDATED IN StudentMarksPercentage');
EXCEPTION
   WHEN no_data_found THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: PERCENTAGE NOT UPDATED');
END updatePercentage;
/
```

```
CREATE OR REPLACE PROCEDURE updatePercentage
IS
BEGIN
    UPDATE StudentSubjectInformation SET
    StudentMarksPercentage = SubjectObtainedMarks/SubjectTotalMarks*100;
    DBMS_OUTPUT.PUT_LINE('PERCENTAGE UPDATED IN StudentMarksPercentage');
EXCEPTION
    WHEN no_data_found THEN
    DBMS_OUTPUT.PUT_LINE('ERROR: PERCENTAGE NOT UPDATED');
END updatePercentage;
/
```

```
Procedure created.
```

- EXECUTE/RUN PROCEDURE
  - BEFORE: (20 ROWS)

SUBJECTOPTED	STUDENTROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	3	100	89	-	1
DS	3	70	55	-	2
DBMS	4	100	90	-	1
ALGO	4	70	60	-	3
ALGO	5	100	86	-	3
DS	5	70	62	-	2
DBMS	1	100	91	-	1
DS	2	80	69	-	2
DBMS	2	70	50	-	1
ALGO	1	70	60	-	3
DBMS	5	100	91	-	1
DS	6	80	55	-	2
DBMS	7	100	82	-	1
ALGO	6	75	52	-	3

- AFTER: (20 rows)
  Exec updatePercentage;

```
EXEC updatePercentage;

SELECT * FROM StudentSubjectInformation;
```

```
Statement processed.
PERCENTAGE UPDATED IN StudentMarksPercentage
```

SUBJECTOPTED	STUDENTROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	3	100	89	89	1
DS	3	70	55	78.57	2
DBMS	4	100	90	90	1
ALGO	4	70	60	85.71	3
ALGO	5	100	86	86	3
DS	5	70	62	88.57	2
DBMS	1	100	91	91	1
DS	2	80	69	86.25	2
DBMS	2	70	50	71.43	1
ALGO	1	70	60	85.71	3

Solving Q10:

Due to the Ambiguity of ScholarshipCategory, updating the category to MERIT / NON-MERIT Category as per the convenience for this question.

IF PERCENTAGE>80 THEN MERIT
ELSE NON-MERIT

- CREATE FUNCTION: to get Percentage of a student using RollNo

```
CREATE OR REPLACE FUNCTION get_percentage(RollNo IN INT)
RETURN DECIMAL
IS
   percentage StudentSubjectInformation.StudentMarksPercentage%TYPE;
BEGIN
   SELECT SUM(StudentMarksPercentage)/Count(StudentRollNo) INTO percentage
   FROM StudentSubjectInformation
   WHERE StudentRollNo = RollNo
   Group By StudentRollNo;
   RETURN percentage;
EXCEPTION
   WHEN no_data_found THEN
      RETURN NULL;
END get_percentage;
```

- CREATE PROCEDURE: to update Scholarship Category of all students

```
CREATE OR REPLACE PROCEDURE updateCategoryAll
IS
       percentage StudentSubjectInformation.StudentMarksPercentage%TYPE;
       sCategory SubjectScholarshipInformation.ScholarshipCategory%TYPE;
BEGIN
   For c in (Select StudentRollNo from StudentBasicInformation)
      LOOP percentage := get_percentage(c.StudentRollNo);
         IF percentage IS NOT NULL THEN
                IF percentage > 80
            THEN
               sCategory := 'MERIT';
                   ELSE
                      sCategory := 'NON-MERIT';
                   END IF;
             UPDATE SubjectScholarshipInformation SET
            ScholarshipCategory = sCategory
            WHERE StudentRollNo = c.StudentRollNo;
                DBMS_OUTPUT.PUT_LINE('CATEGORY UPDATED AS PER PERFORMANCE');
         ELSE
            DBMS_OUTPUT.PUT_LINE('NO RECORD IN PERCENTAGE TABLE FOR THIS ROLL NO.');
         END IF;
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('ScholarshipCategory UPDATED IN StudentScholarshipInformation');
EXCEPTION
       WHEN no_data_found
       THEN
                DBMS_OUTPUT.PUT_LINE('ERROR: ScholarshipCategory NOT UPDATED');
END updateCategoryAll;
```

- EXEC updateCategoryAll;

```
4   exec updateCategoryAll;
```

```
Statement processed.
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
CATEGORY UPDATED AS PER PERFORMANCE
ScholarshipCategory UPDATED IN StudentScholarshipInformation
```

- SELECT * FROM SubjectScholarshipInformation;

```
1
2   SELECT * FROM SubjectScholarshipInformation;
3
4   -- exec updateCategoryAll;
```

SCHOLARSHIPID	STUDENTROLLNO	SCHOLARSHIPNAME	SCHOLARSHIPDESCRIPTION	SCHOLARSHIPAMOUNT	SCHOLARSHIPCATEGORY	SCHOLARSHIPSTARTDATE	SCHOLARSHIPENDDATE
1	1	Indira Gandhi Scholarship Scheme	The scheme is for a single girl child who has taken admission in regular, full-time	36200	MERIT	11-AUG-19	11-AUG-20
2	2	PG Merit Scholarship for University Rank Holder	Candidate should be the first or second rank holder at the UG level	3100	NON-MERIT	09-AUG-18	09-SEP-20
3	4	Special Scholarship Scheme Ishan Uday for NER	The annual income of the family should not be more than Rs.4.5 Lakh.	7200	MERIT	01-SEP-17	01-AUG-18
4	3	Indira Gandhi Scholarship Scheme	The scheme is applicable to such a single girl child who has taken admission in regular	36200	MERIT	11-JUN-18	11-JUN-19

Solving Q11:

- Query:

```
CREATE OR REPLACE VIEW StudentDetailsBalance AS
        SELECT StudentAdmissionPaymentDetails.StudentRollNo, StudentName, StudentSurName,
                StudentSex, StudentBdate, StudentMobile, StudentAddress, AmountBalance
        FROM StudentBasicInformation INNER JOIN StudentAdmissionPaymentDetails
        ON StudentBasicInformation.StudentRollNo = StudentAdmissionPaymentDetails.StudentRollNo;
```

- Output

```
-- View which shows balance amount to be paid by the student along with the student detailed information
CREATE OR REPLACE VIEW StudentDetailsBalance AS
    SELECT StudentAdmissionPaymentDetails.StudentRollNo, StudentName, StudentSurName,
           StudentSex, StudentBdate, StudentMobile, StudentAddress, AmountBalance
    FROM StudentBasicInformation INNER JOIN StudentAdmissionPaymentDetails
    ON StudentBasicInformation.StudentRollNo = StudentAdmissionPaymentDetails.StudentRollNo;

SELECT * FROM StudentDetailsBalance;
```

STUDENTROLLNO	STUDENTNAME	STUDENTSURNAME	STUDENTSEX	STUDENTBDATE	STUDENTMOBILE	STUDENTADDRESS	AMOUNTBALANCE
1	Ram	Gopal	M	12-JAN-16	9812347816	123/5 Bhopal,MP	2000
1	Ram	Gopal	M	12-JAN-16	9812347816	123/5 Bhopal,MP	3000
2	Ramya	Goel	F	01-FEB-01	8818847816	House-995 Ashok Vihar,Delhi	1970
3	Sanya	Agarwal	F	01-FEB-94	7812847816	House-11 Preet Vihar,Gujarat	0
2	Ramya	Goel	F	01-FEB-01	8818847816	House-995 Ashok Vihar,Delhi	1970
3	Sanya	Agarwal	F	01-FEB-94	7812847816	House-11 Preet Vihar,Gujarat	0
4	Deepak	Chauhan	M	11-AUG-95	7812819116	812/11 Ram Nagar,Gujarat	1970
4	Deepak	Chauhan	M	11-AUG-95	7812819116	812/11 Ram Nagar,Gujarat	2000
5	Nishant	Chauhan	M	01-OCT-93	9912819116	11/4 Ramya Building,Mumbai	0
6	Priyanka	Chopra	F	16-MAR-98	9910019116	Flat-101 Santok Tower,Mumbai	1990

Download CSV

10 rows selected.

Solving Q12:

- Query:

```
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo NOT IN
(
    SELECT StudentRollNo FROM SubjectScholarshipInformation
)
```

OR

```
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo NOT IN
(
    SELECT si.StudentRollNo
    FROM StudentBasicInformation si, SubjectScholarshipInformation ss
    WHERE si.StudentRollNo=ss.StudentRollNo
)
```

- Output:

```
-- Students not got the scholarship
SELECT * FROM StudentBasicInformation
WHERE StudentRollNo NOT IN
(
    SELECT StudentRollNo FROM SubjectScholarshipInformation
)
```

STUDENTNAME	STUDENTSURNAME	STUDENTROLLNO	STUDENTADDRESS	STUDENTBDATE	STUDENTMOBILE	STUDENTSEX
Ramya	Goel	2	House-995 Ashok Vihar,Delhi	01-FEB-01	8818847816	F

Download CSV

Solving Q13:

PROCEDURE: NO RETURN

- CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE getBalanceAmount (RollNo in INT)
IS
        balanceAmount StudentAdmissionPaymentDetails.AmountBalance%TYPE;
BEGIN
        SELECT SUM(AmountBalance) INTO balanceAmount
        FROM StudentAdmissionPaymentDetails
        WHERE StudentRollNo = RollNo
        GROUP BY StudentRollNo;

        DBMS_OUTPUT.PUT_LINE(RollNo || ' : ' || balanceAmount);
EXCEPTION
        WHEN no_data_found THEN
                DBMS_OUTPUT.PUT_LINE(RollNo || ' : Not Found');
END getBalanceAmount;
```

```
52  CREATE OR REPLACE PROCEDURE getBalanceAmount (RollNo in INT)
53  IS
54          balanceAmount StudentAdmissionPaymentDetails.AmountBalance%TYPE;
55  BEGIN
56          SELECT SUM(AmountBalance) INTO balanceAmount
57      FROM StudentAdmissionPaymentDetails
58          WHERE StudentRollNo = RollNo
59          GROUP BY StudentRollNo;
60          DBMS_OUTPUT.PUT_LINE(RollNo || ' : ' || balanceAmount);
61  EXCEPTION
62          WHEN no_data_found THEN
63                  DBMS_OUTPUT.PUT_LINE(RollNo || ' : Not Found');
64  END getBalanceAmount;
```

```
Procedure created.
```

- EXECUTE/RUN PROCEDURE
  - exec getBalanceAmount(1) ; exec getBalanceAmount(5) ; exec getBalanceAmount(10) ;

```
51
52  exec getBalanceAmount(1) ;
53  exec getBalanceAmount(5) ;
54  exec getBalanceAmount(10) ;
55
```

```
Statement processed.
1 : 5000

Statement processed.
5 : 0

Statement processed.
10 : Not Found
```

- BEGIN
  FOR c IN
  (
  SELECT StudentRollNo
  FROM STUDENTBASICINFORMATION
)LOOP
  getBalanceAmount(c.StudentRollNo);
  END LOOP;
  END;

```
BEGIN
  FOR c IN (SELECT StudentRollNo FROM STUDENTBASICINFORMATION ) LOOP
        getBalanceAmount(c.StudentRollNo);
  END LOOP;
END;
```

```
Statement processed.
1 : 5000
2 : 3940
3 : 0
4 : 3970
5 : 0
6 : 1990
7 : Not Found
8 : Not Found
9 : Not Found
10 : Not Found
```

- BEGIN
  FOR c IN
  (
  SELECT StudentRollNo
  FROM STUDENTBASICINFORMATION
  WHERE StudentRollNo IN (3,6,9,2)
)LOOP
  getBalanceAmount(c.StudentRollNo);
  END LOOP;
  END;

```
BEGIN
  FOR c IN (SELECT StudentRollNo FROM STUDENTBASICINFORMATION WHERE StudentRollNo IN (3,6,9,2) ) LOOP
        getBalanceAmount(c.StudentRollNo);
  END LOOP;
END;
```

```
Statement processed.
2 : 3940
3 : 0
6 : 1990
9 : Not Found
```

PROCEDURE: NO RETURN

- CREATE PROCEDURE

```
CREATE OR REPLACE FUNCTION getBalanceAmount1 (RollNo in INT)
RETURN StudentAdmissionPaymentDetails.AmountBalance%TYPE
IS
      balanceAmount StudentAdmissionPaymentDetails.AmountBalance%TYPE;
BEGIN
   SELECT SUM(AmountBalance) INTO balanceAmount
   FROM StudentAdmissionPaymentDetails
   WHERE StudentRollNo = RollNo
   GROUP BY StudentRollNo;
   RETURN balanceAmount;
EXCEPTION
   WHEN no_data_found THEN
     return null;
END getBalanceAmount1;
/
```

```
CREATE OR REPLACE FUNCTION getBalanceAmount1 (RollNo in INT)
RETURN StudentAdmissionPaymentDetails.AmountBalance%TYPE
IS
    balanceAmount StudentAdmissionPaymentDetails.AmountBalance%TYPE;
BEGIN
    SELECT SUM(AmountBalance) INTO balanceAmount
    FROM StudentAdmissionPaymentDetails
    WHERE StudentRollNo = RollNo
    GROUP BY StudentRollNo;
    RETURN balanceAmount;
EXCEPTION
    WHEN no_data_found THEN
        return null;
END getBalanceAmount1;
/
```

```
Function created.
```

- EXECUTE/RUN PROCEDURE

  - SELECT DISTINCT(StudentRollNo),getBalanceAmount1(StudentRollNo) AS balanceAmount
    FROM StudentAdmissionPaymentDetails
    ORDER BY StudentRollNo;

```
SELECT DISTINCT(StudentRollNo),getBalanceAmount1(StudentRollNo) AS balanceAmount
FROM StudentAdmissionPaymentDetails
ORDER BY StudentRollNo
```

STUDENTROLLNO	BALANCEAMOUNT
1	5000
2	3940
3	0
4	3970
5	0
6	1990

Download CSV
6 rows selected.

  - SELECT StudentRollNo,getBalanceAmount1(StudentRollNo) AS balanceAmount
    FROM StudentBasicInformation
    ORDER BY StudentRollNo;

```
select StudentRollNo,getBalanceAmount1(StudentRollNo) AS balanceAmount
FROM StudentBasicInformation
order by StudentRollNo;
```

STUDENTROLLNO	BALANCEAMOUNT
1	5000
2	3940
3	0
4	3970
5	0
6	1990
7	-
8	-
9	-
10	-

Download CSV
10 rows selected.

Solving Q14:

- Query:

```
SELECT * FROM
(
        SELECT * FROM StudentSubjectInformation
        ORDER BY STUDENTMARKSPERCENTAGE DESC
)
WHERE ROWNUM <= 5 ;
```

- Output:

```
-- SELECT * FROM StudentSubjectInformation ORDER BY STUDENTMARKSPERCENTAGE DESC;
SELECT * FROM
(
    SELECT * FROM StudentSubjectInformation ORDER BY STUDENTMARKSPERCENTAGE DESC
)
WHERE ROWNUM <= 5 ;
```

SUBJECTOPTED	STUDENTROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	1	100	91	91	1
DBMS	4	100	90	90	1
DBMS	3	100	89	89	1
DS	5	70	62	88.57	2
DS	2	80	69	86.25	2

Download CSV

5 rows selected.

Solving Q15:

I. INNER JOIN

To show balance amount to be paid by the student along with the student detailed information

- Query

```
SELECT T1.StudentRollNo, StudentName, StudentSurName,
        StudentSex, StudentBdate, StudentMobile, StudentAddress, AmountBalance
FROM StudentBasicInformation T1 INNER JOIN StudentAdmissionPaymentDetails T2
ON T1.StudentRollNo = T2.StudentRollNo;
```

- Output

```
SELECT T1.StudentRollNo, StudentName, StudentSurName,
       StudentSex, StudentBdate, StudentMobile, StudentAddress, AmountBalance
FROM StudentBasicInformation T1 INNER JOIN StudentAdmissionPaymentDetails T2
ON T1.StudentRollNo = T2.StudentRollNo;
```

STUDENTROLLNO	STUDENTNAME	STUDENTSURNAME	STUDENTSEX	STUDENTBDATE	STUDENTMOBILE	STUDENTADDRESS	AMOUNTBALANCE
1	Ram	Gopal	M	12-JAN-16	9812347816	123/5 Bhopal,MP	2000
1	Ram	Gopal	M	12-JAN-16	9812347816	123/5 Bhopal,MP	3000
2	Ramya	Goel	F	01-FEB-01	8818847816	House-995 Ashok Vihar,Delhi	1970
3	Sanya	Agarwal	F	01-FEB-94	7812847816	House-11 Preet Vihar,Gujarat	0
2	Ramya	Goel	F	01-FEB-01	8818847816	House-995 Ashok Vihar,Delhi	1970
3	Sanya	Agarwal	F	01-FEB-94	7812847816	House-11 Preet Vihar,Gujarat	0
4	Deepak	Chauhan	M	11-AUG-95	7812819116	812/11 Ram Nagar,Gujarat	1970
4	Deepak	Chauhan	M	11-AUG-95	7812819116	812/11 Ram Nagar,Gujarat	2000
5	Nishant	Chauhan	M	01-OCT-93	9912819116	11/4 Ramya Building,Mumbai	0
6	Priyanka	Chopra	F	16-MAR-98	9910019116	Flat-101 Santok Tower,Mumbai	1990

Download CSV

10 rows selected.

II. LEFT JOIN

To Show RollNo, Name and Mobile No. of Students who have not made any Admission Payement

- Query

```
SELECT T1.StudentRollNo, StudentName || ' ' || StudentSurname as StudentFullName,
StudentMobile
FROM StudentBasicInformation T1 LEFT JOIN StudentAdmissionPaymentDetails T2
ON T1.StudentRollNo = T2.StudentRollNo WHERE T2.TransactionId IS NULL;
```

- Output

```
SELECT T1.StudentRollNo, StudentName || ' ' || StudentSurname as StudentFullName, StudentMobile
FROM StudentBasicInformation T1 LEFT JOIN StudentAdmissionPaymentDetails T2
ON T1.StudentRollNo = T2.StudentRollNo
WHERE T2.TransactionId IS NULL;
```

STUDENTROLLNO	STUDENTFULLNAME	STUDENTMOBILE
7	Virat Kohli	9912800116
8	M.S Dhoni	9912800777
10	Saadhvi Mehra	9900822116
9	Alia Bhatt	9900800116

Download CSV

4 rows selected.

III. NATURAL JOIN

Retrieve the percentage of the students along with students detailed information who has scored the highest percentage along with availing the maximum scholarship amount

- Query

```
SELECT * FROM
(
  (
    SELECT StudentRollNo, SUM(StudentMarksPercentage)/COUNT(StudentRollNo) AS
Percentage
    FROM StudentSubjectInformation
    GROUP BY StudentRollNo
  )
  NATURAL JOIN
  (
    SELECT StudentRollNo,SUM(ScholarshipAmount) AS TotalAmount
    FROM SubjectScholarshipInformation
    GROUP BY StudentRollNo
  )
) ORDER BY Percentage DESC, TotalAmount DESC;
```

- Output

STUDENTROLLNO	PERCENTAGE	TOTALAMOUNT
9	88.66	5000
5	88.5233333333333333333333333333333333333	3100
1	88.355	40200
4	87.855	7200
3	83.785	35000
7	82	8000
10	82	2000
8	78.25	3100
6	69.04	8000

Download CSV

9 rows selected.

Solving Q16:

I. DROP

DROP statement is a Data Definition Language(DDL) Command which is used to delete existing database objects. It can be used to delete databases, tables, views, triggers, etc.

- Syntax:

  DROP object object_name

- Examples:
  - DROP TABLE Employees;
    This query will remove the whole table Employees from the database.
  - DROP DATABASE Company;
    This query will delete the database Company.

II.DROP

The DELETE statement in SQL is a Data Manipulation Language(DML) Command. It is used to delete existing records from an existing table. We can delete a single record or multiple records depending on the condition specified in the query.

- The DELETE statement scans every row before deleting it.
  Thus, it is slower as compared to TRUNCATE command. If we want to delete all the records of a table, it is preferable to use TRUNCATE in place of DELETE as the former is faster than the latter.

- Syntax:

  DELETE FROM table_name [WHERE conditions];

- Examples:
  - DELETE FROM Employees WHERE Emp_Id = 7;
    This query will delete the record(s) from Employees table where field Emp_Id has a value 7.

III.TRUNCATE

TRUNCATE Command is a Data Definition Language operation. It is used to remove all the records from a table. It deletes all the records from an existing table but not the table itself.

- The structure or schema of the table is preserved.

- Syntax:

  TRUNCATE TABLE table_name;

- Examples:
  - TRUNCATE TABLE Employees;
    This query will remove all the records from the table Employees.

- Truncate statement is equivalent to DELETE operation without a WHERE clause. The truncate command removes the records from a table without scanning it. This is why it is faster than the DELETE statement.

Solving Q17:

- Query

```
SELECT ScholarshipCategory, COUNT(ScholarshipCategory) AS Studentcount
FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory ;
```

- Output

```
SELECT ScholarshipCategory, COUNT(ScholarshipCategory) AS Studentcount
FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory ;
```

SCHOLARSHIPCATEGORY	STUDENTCOUNT
ST	1
OBC	1
HANDICAPPED	1
ALL	4
SC	3

Download CSV

5 rows selected.

Solving Q18:

- Query

```
SELECT ScholarshipCategory, COUNT(ScholarshipCategory) AS highestScholarishpCount
FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory
HAVING COUNT(ScholarshipCategory) = (
   SELECT MAX( mycount ) FROM
   (
      SELECT ScholarshipCategory, COUNT(*) AS mycount
      FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory
   )
);
```

- Output

```
SELECT ScholarshipCategory, COUNT(ScholarshipCategory) AS highestScholarishpCount
FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory
HAVING COUNT(ScholarshipCategory)= (
    SELECT MAX( mycount ) FROM
    (
        SELECT ScholarshipCategory, COUNT(*) AS mycount
        FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory
    )
);
```

SCHOLARSHIPCATEGORY	HIGHESTSCHOLARISHPCOUNT
ALL	4

Download CSV

Solving Q19:

For this inserted more records into StudentSubjectInformation Table and thus it becomes:

SUBJECTOPTED	ROLLNO	SUBJECTTOTALMARKS	SUBJECTOBTAINEDMARKS	STUDENTMARKSPERCENTAGE	SEMESTER
DBMS	1	100	91	91	1
ALGO	1	70	60	85.71	3
DBMS	2	70	50	71.43	1
DS	2	80	69	86.25	2
DBMS	3	100	89	89	1
DS	3	70	55	78.57	2
ALGO	4	70	60	85.71	3
DBMS	4	100	90	90	1
ALGO	5	100	86	86	3
DBMS	5	100	91	91	1
DS	5	70	62	88.57	2
ALGO	6	75	52	69.33	3
DS	6	80	55	68.75	2
DBMS	7	100	82	82	1
ALGO	8	100	86	86	3
DS	8	70	42	60	2
DBMS	8	80	71	88.75	1
DS	9	80	79	98.75	2
DBMS	9	70	55	78.57	1
ALGO	10	100	82	82	3

- Query

```
SELECT * FROM
(
   (
      SELECT StudentRollNo, SUM(StudentMarksPercentage)/COUNT(StudentRollNo) AS Percentage
      FROM StudentSubjectInformation
      GROUP BY StudentRollNo
   )
   NATURAL JOIN
   (
      SELECT StudentRollNo,SUM(ScholarshipAmount) AS TotalAmount
      FROM SubjectScholarshipInformation
      GROUP BY StudentRollNo
   )
) ORDER BY Percentage DESC, TotalAmount DESC;
```

- Output

STUDENTROLLNO	PERCENTAGE	TOTALAMOUNT
9	88.66	5000
5	88.5233333333333333333333333333333333333	3100
1	88.355	40200
4	87.855	7200
3	83.785	35000
7	82	8000
10	82	2000
8	78.25	3100
6	69.04	8000

Download CSV

9 rows selected.

Solving Q20:

1. TRIGGERS:

- Trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete)
- It can execute automatically based on the events.
- It cannot take input as parameter
- we can't use transaction statements inside a trigger

2. STORED PROCEDURES

- Stored procedures are a piece of the code in written in PL/SQL to do some specific task.
- It can be invoked explicitly by the user
- It can take input as a parameter
- We can use transaction statements like begin transaction, commit transaction, and rollback inside a stored procedure.

3. VIEWS

- A view is a virtual table based on the result-set of an SQL statement.
- It contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.
- Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_nameWHERE condition;
```

4. FUNCTIONS

- There are two types of SQL functions, aggregate functions, and scalar(non-aggregate) functions.
- Aggregate functions operate on many records and produce a summary, works with GROUP BY
- Non-aggregate functions operate on each record independently. There are so many built-in functions in SQL to do various calculations on data.
- Some Aggregate functions are – SQL Count function, SQL Sum function, SQL Avg function, SQL Max function, SQL Min function
- Some Arithmetic functions are - abs(), ceil(), floor(), exp(), ln(), mod(), power(), sqrt()

Submitted By:
Disha Surana
(disha.surana@accolitedigital.com)