ASSIGNMENT DBMS(SQL)

Submitted by Pradeep kumar

[Pradeep.kumar@accolite\_digital.com](mailto:Pradeep.kumar@accolite_digital.com)

\*note : we use oracle live sql server to create our student database and all the tables

**Ques2: CREATING ALL THE TABLES:**

**i)**

CREATE TABLE StudentBasicInformation

( StudentName VARCHAR(10) NOT NULL,

StudentSurName VARCHAR(10),

StudentRollNo INT,

StudentAddress VARCHAR(20),

StudentBdate DATE,

StudentSex CHAR(1),

StudentAge INT,

PRIMARY KEY (StudentRollNo))

**ii)**

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)

)

**iii)**

create table StudentSubjectInformation

(

SubjectOpted CHAR(7),

StudentRollNo INT,

SubjectTotalMarks INT,

SubjectObtainedMarks INT,

StudentMarksPercentage NUMERIC(4,2),

Semester INT,

PRIMARY KEY (SubjectOpted,StudentRollNo),

FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)

)

**iv)**

CREATE TABLE SubjectScholarshipInformation

(

ScholarshipId NUMBER GENERATED BY DEFAULT AS IDENTITY,

StudentRollNo INT,

ScholarshipName VARCHAR(20),

ScholarshipDescription VARCHAR(30),

ScholarshipAmount DECIMAL(8,2),

ScholarshipCategory VARCHAR(10),

ScholarshipStartDate DATE,

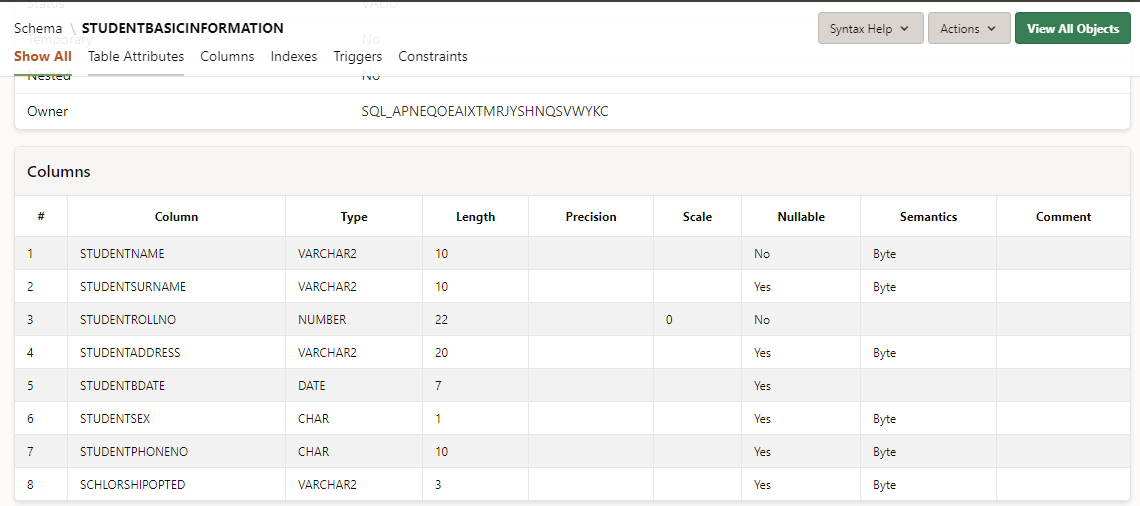
ScholarshipEndDate DATE,

PRIMARY KEY (ScholarshipId),

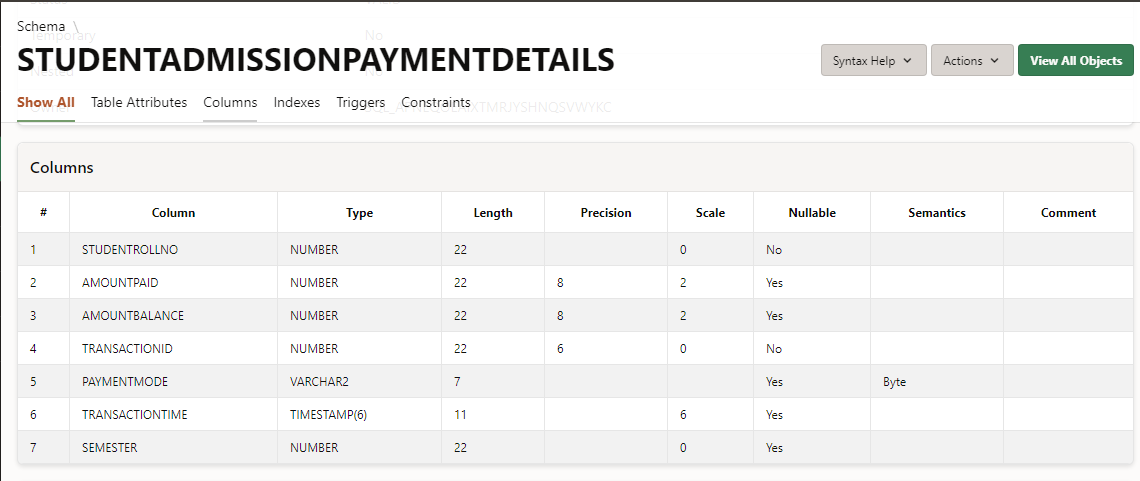
FOREIGN KEY (StudentRollNo) references StudentBasicInformation(StudentRollNo)

)

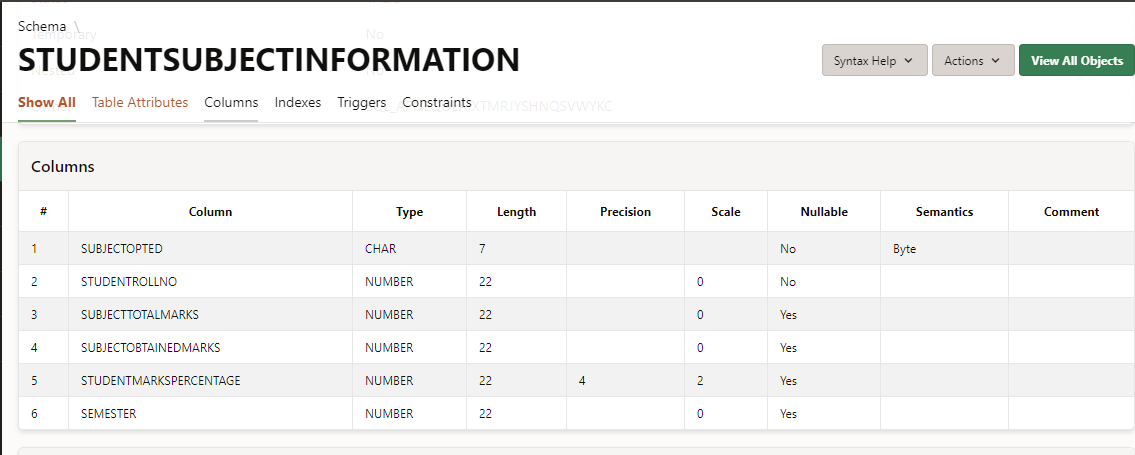
**THEIR RESPECTIVE SCHEMAS ARE:**



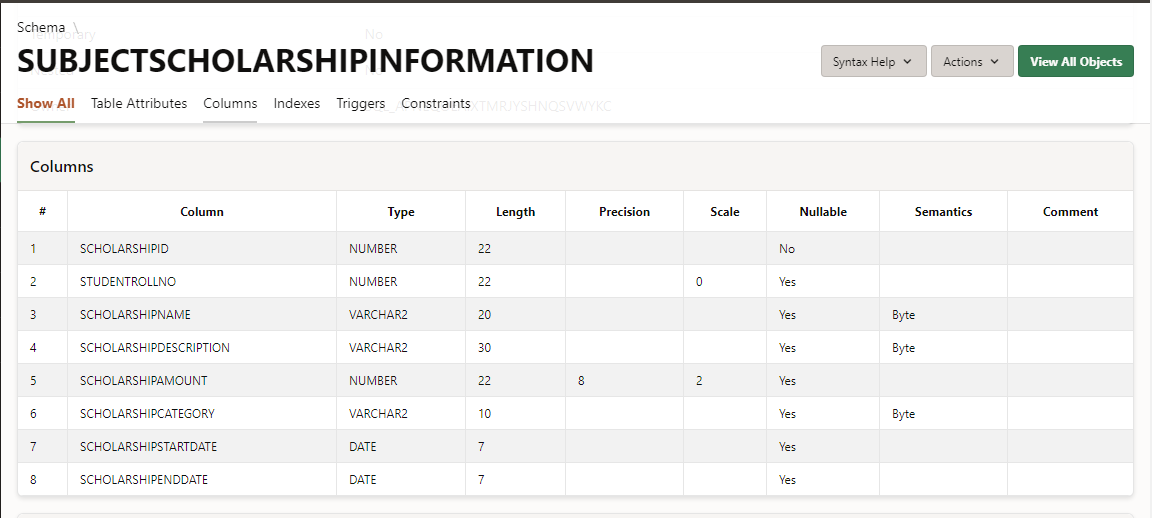
II.



iii)

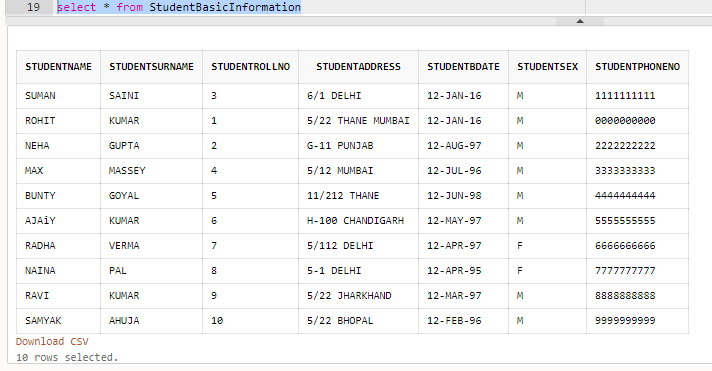


iv)

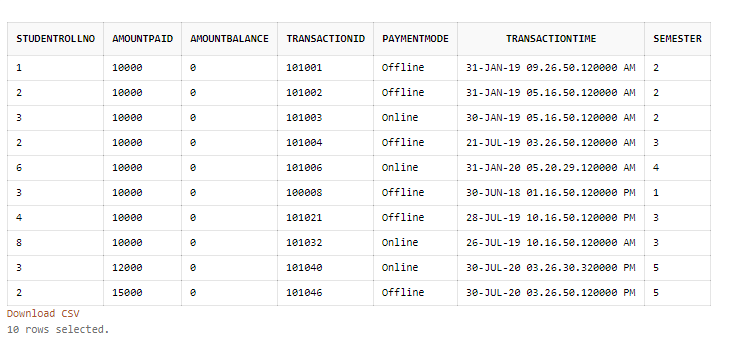


**Ques3and4:**

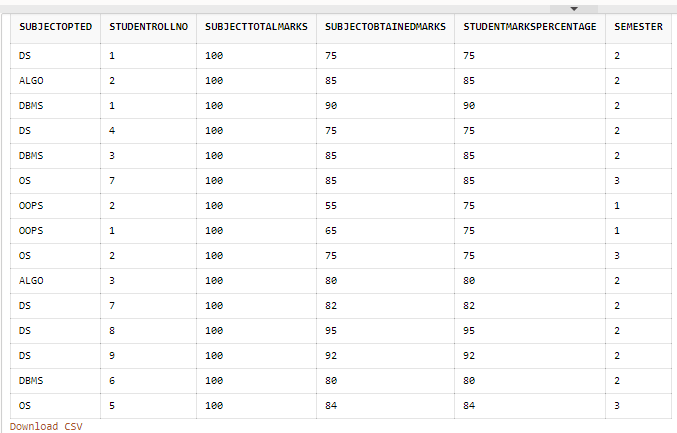
StudentBasicinformation



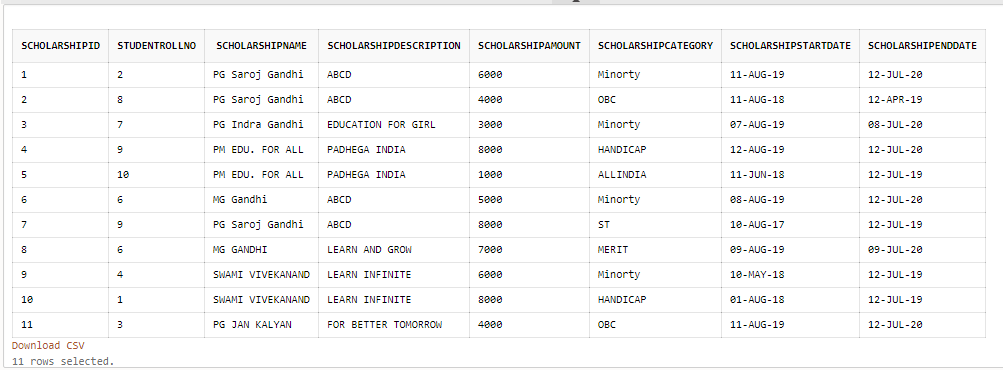
StudentAdmissionPaymentDetails



**StudentSubjectInformation**



**SubjectScholarshipInformation**

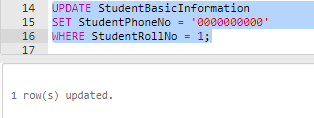


Ques5: Updated the table StudentBasicInformation by adding one more column

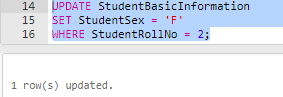
1. update StudentBasicInformation

set StudentScholarshipType ='YES' where StudentRollNo>0; // added one more column

1. – Here updated Phone number of one student.



1. Updated student sex information



1. Update column name (rename) by using alter command

alter table StudentBasicInformation

rename column StudentScholarshipType to SchlorshipOpted;

1. Update Student name

UPDATE StudentBasicInformation

SET StudentName = 'AJAY'

WHERE StudentRollNo = 6

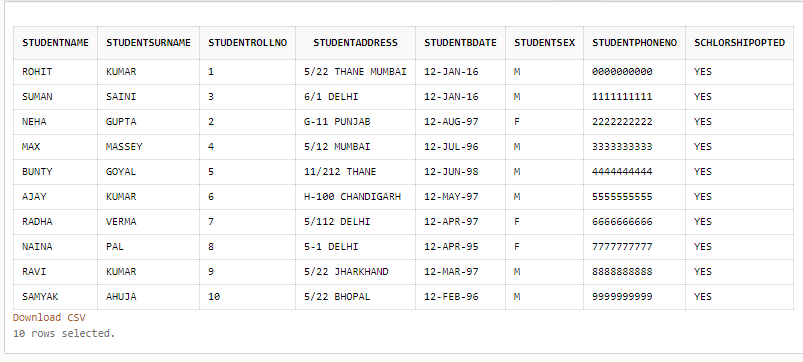
#snapshot:

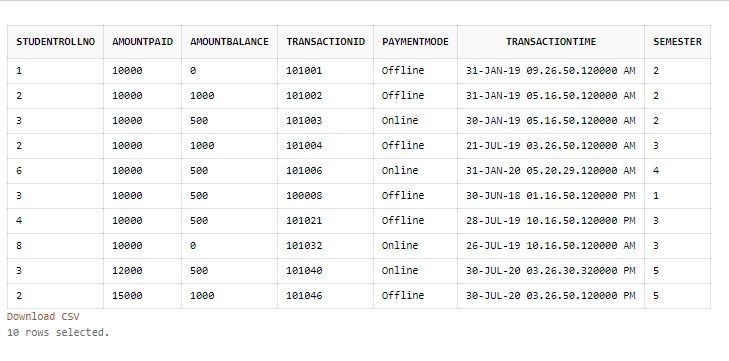


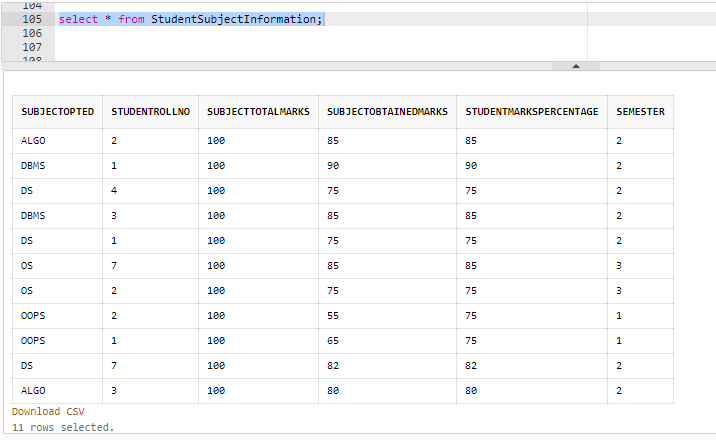


1. Update amount balance column of StudentAdmissionPaymentDetails

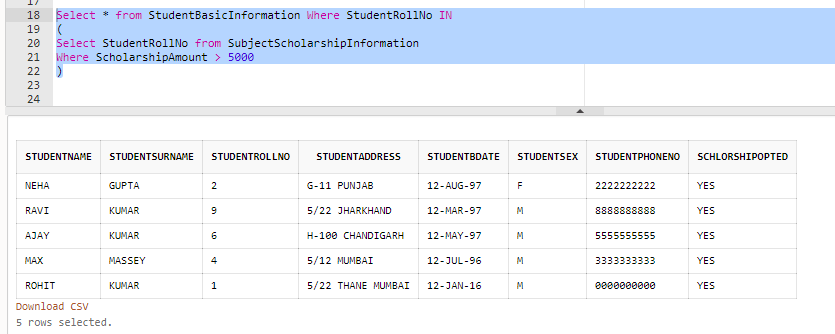
**Ques6: snaps**



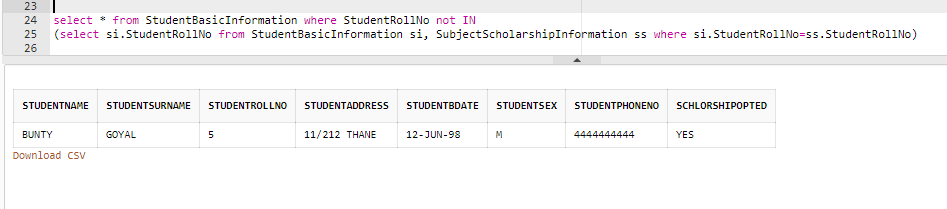




Ques7 :

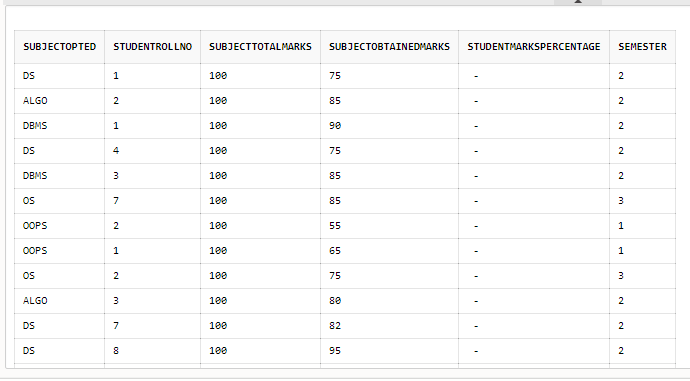


**Ques8:**



**Ques9**

BY USING UPDATE COMMAND FIRST I SET ALL THE StudentMarksPercentage to NULL



THEN USE THE PROCEDURE TO FILL THE StudenteMarksInformation

Here the sql query for this :

CREATE OR REPLACE PROCEDURE fillPercentage

IS

BEGIN

UPDATE StudentSubjectInformation SET

StudentMarksPercentage = SubjectObtainedMarks/SubjectTotalMarks\*100;

DBMS\_OUTPUT.PUT\_LINE('PERCENTAGE FILLED IN StudentMarksPercentage');

EXCEPTION

WHEN no\_data\_found THEN

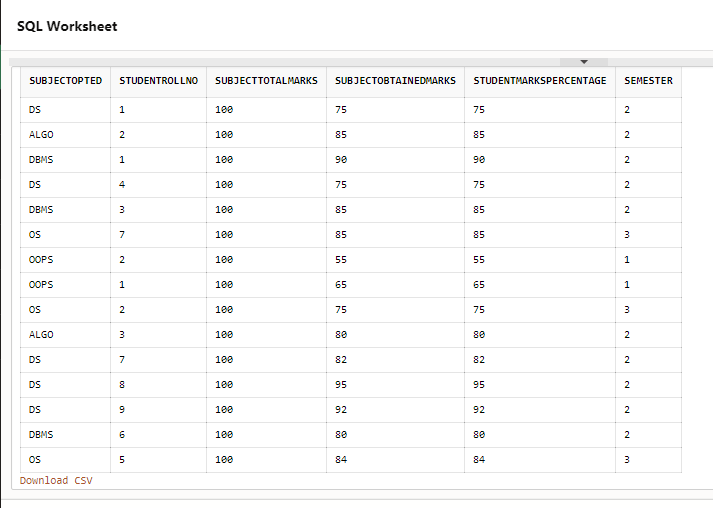
DBMS\_OUTPUT.PUT\_LINE('ERROR: PERCENTAGE NOT FILLED');

END updatePercentage;

/

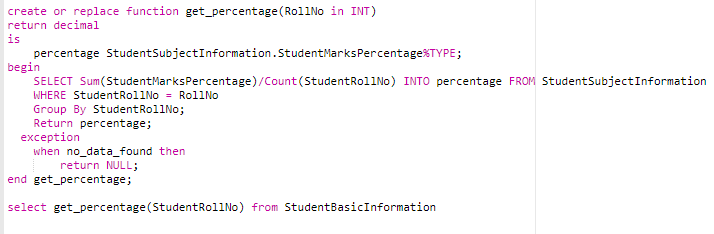
exec fillPercentage(); // to call the procedure

output after using query the table information changes and filled with StudentMarksPercentage



**Ques10**

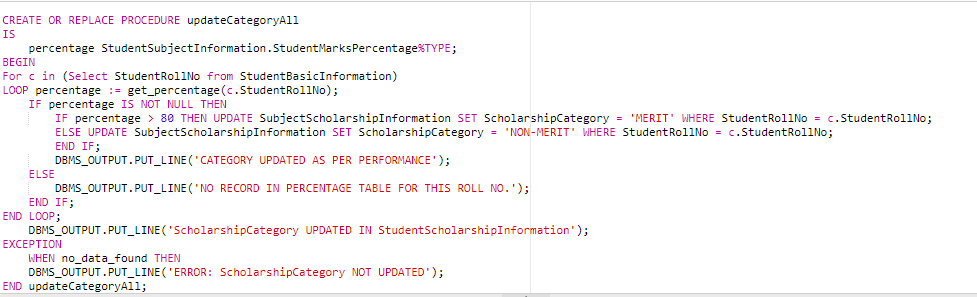
First create the getPercentage function to get the averagepercentagemarks



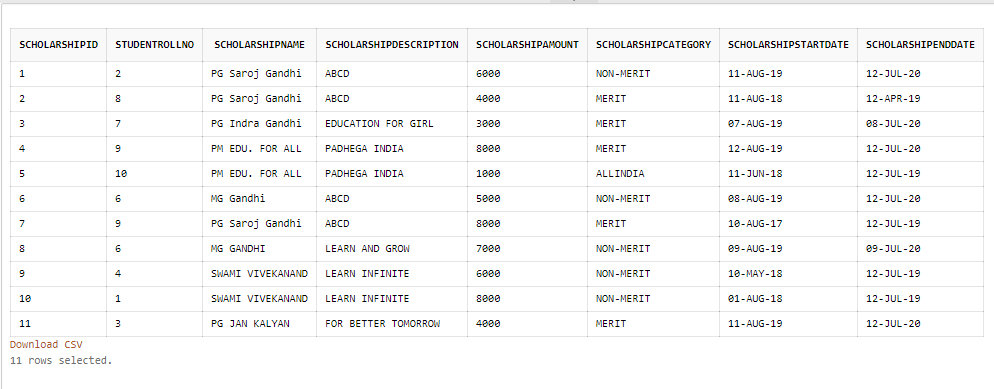
* The execute the procedure for all the students who scored greater than 80

Student who got more than 80 percentage we assign the acholarshipcategory to merit

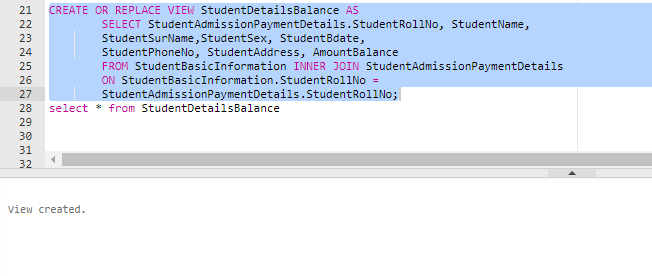
Else non merit



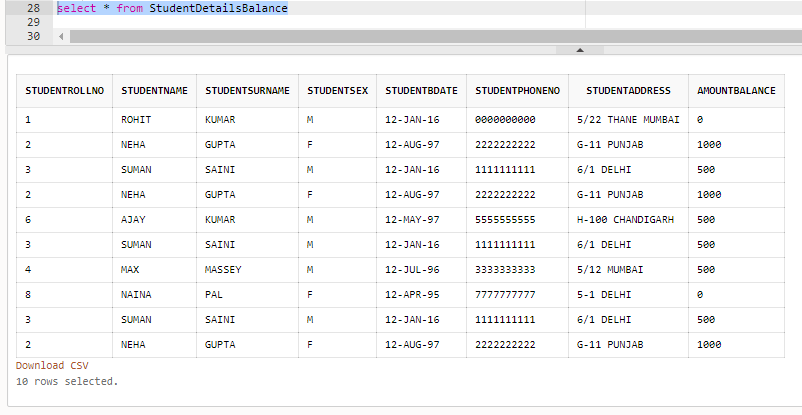
After select \* from SubjectScholarshipInformatiom we get the updated table as below:



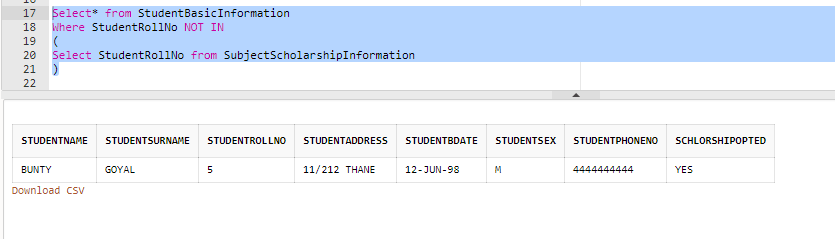
**Ques11**



View created is below



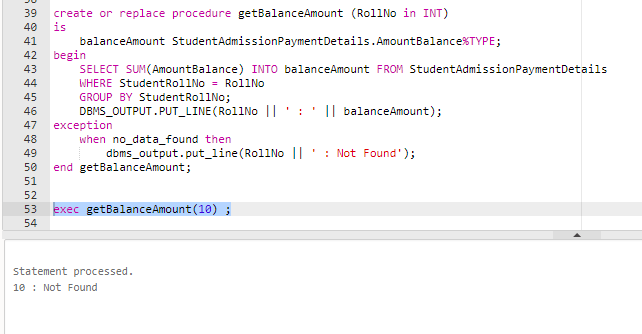
**Ques 12:**



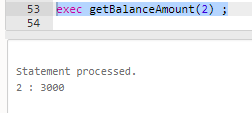
**Ques13**

HERE we create procedure to get the AmountBalance for student by using their rollno

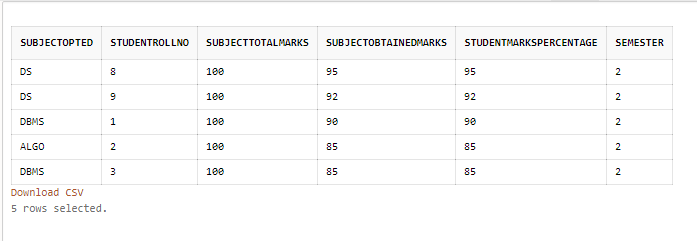
1. In first exec we get no pending due of dtudent whose rolno is 10
2. In second exec we get pending amount details



ii)



**Ques14:**



**Ques15**

1. **LEFT JOIN**

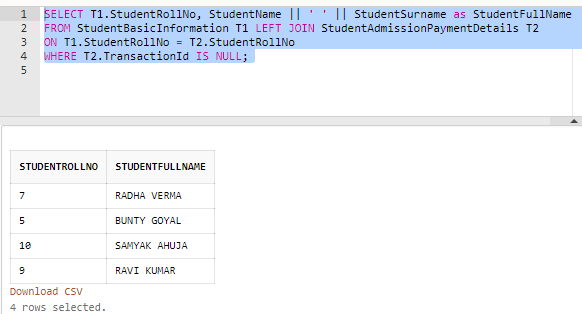
**To Show RollNo, Name and Mobile No. of Students who have not made any Admission Payement**

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;



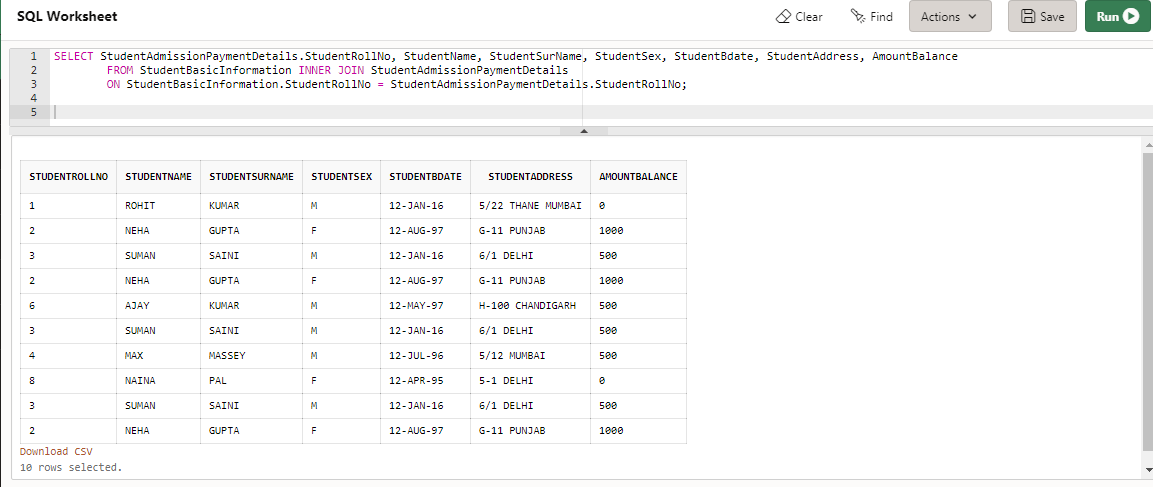
1. INNER JOIN

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

SELECT StudentAdmissionPaymentDetails.StudentRollNo, StudentName, StudentSurName, StudentSex, StudentBdate, StudentMobile, StudentAddress, AmountBalance

     FROM StudentBasicInformation INNER JOIN StudentAdmissionPaymentDetails

     ON StudentBasicInformation.StudentRollNo = StudentAdmissionPaymentDetails.StudentRollNo;



1. 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

SELECT \* FROM

(

     (

SELECT StudentRollNo, SUM(StudentMarksPercentage)/COUNT(StudentRollNo) AS Percentage

         FROM StudentSubjectInformation

         GROUP BY StudentRollNo

     )

     NATURAL JOIN

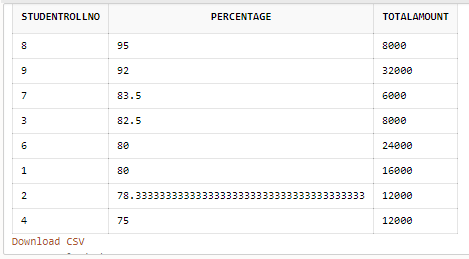
     (

SELECT StudentRollNo,SUM(ScholarshipAmount) AS TotalAmount FROM SubjectScholarshipInformation

         GROUP BY StudentRollNo

     )

) ORDER BY Percentage DESC, TotalAmount DESC;



**Ques16**

**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.

DROP object object\_name

DROP TABLE Employees;

This query will remove the whole table Employees from the database.

DROP DATABASE Company;

This query will delete the database Company.

**DELETE**

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.

DELETE FROM table\_name [WHERE conditions];

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.

**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.

TRUNCATE TABLE table\_name;

TRUNCATE TABLE Employees;

This query will remove all the records from the table Employees.

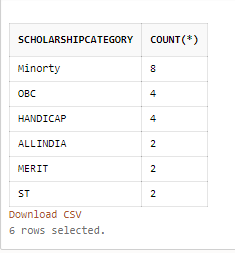
Truncate statement is equivalent to DELETE operation without a WHERE clause. The truncate command removes the records from a table without scanning it. This is why it is faster than the DELETE statement.

**Ques17**

select ScholarshipCategory,count(\*) from SubjectScholarshipInformation

group by ScholarshipCategory

order by count(\*) desc;



**Ques18**

select ScholarshipCategory, count(ScholarshipCategory) as highestNoOfScholarishp from SubjectScholarshipInformation

group by ScholarshipCategory

having count(ScholarshipCategory)=

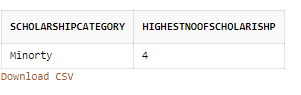
(

Select MAX( mycount ) FROM(

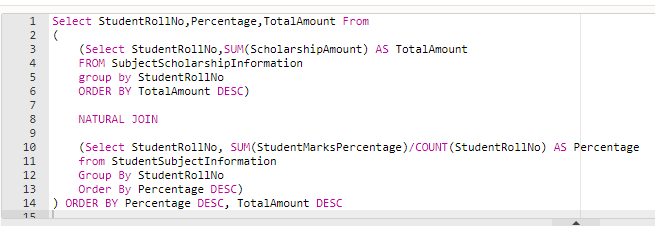
select ScholarshipCategory, count(\*) as mycount from SubjectScholarshipInformation

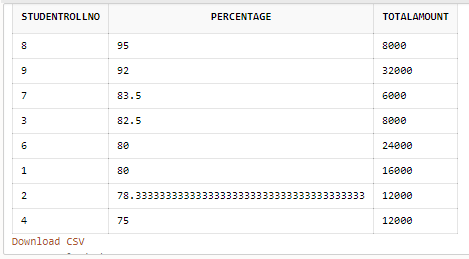
group by ScholarshipCategory)

);



**Ques19:**





**Ques20**

**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 can not take input as parameter
* we can't use transaction statements inside a trigger

**STORED PROCEDURES**

* Stored procedures are a pieces 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.

**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.

CREATE VIEW *view\_name* AS

SELECT *column1*, *column2*, ...

FROM table\_name

WHERE condition;

**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 Avg function](https://www.w3resource.com/sql/aggregate-functions/avg-function.php), [SQL Max function](https://www.w3resource.com/sql/aggregate-functions/max-function.php), [SQL Min function](https://www.w3resource.com/sql/aggregate-functions/min-function.php)
* Some Arithmetic functions are -  [floor()](https://www.w3resource.com/sql/arithmetic-functions/floor-wth-positive-value.php), [exp()](https://www.w3resource.com/sql/arithmetic-functions/exp.php), [ln()](https://www.w3resource.com/sql/arithmetic-functions/ln.php), [mod()](https://www.w3resource.com/sql/arithmetic-functions/mod.php), [power()](https://www.w3resource.com/sql/arithmetic-functions/power.php), [sqrt()](https://www.w3resource.com/sql/arithmetic-functions/sqrt.php)