ABC Convocation Management System

Submitted By-

Id	Name	Contribution	Percentage
20-42143-1	Sudipta Saha	User Interface, Table creation, Data Insertion, Query writing, Relational Algebra, PL/SQL(Cursor, Trigger, Package, Record)	30%
20-42172-1	Himi Roy	Introduction, Project Proposal, Scenario Description, Conclusion	15%
20-42174-1	Ajoy Roy	Normalization, Schema Diagram, Activity Diagram, PL/SQL(Function, Procedure)	30%
18-38421-2	Mahin Al Mukit	Class Diagram, Use Case Diagram, ER Diagram	25%

Course Name: Advance Database Management System

Section: [B]

Table of Contents

Introduction

Convocation ceremonies are significant milestones in the academic journey, where graduates are celebrated for their accomplishments and achievements. However, ABC is a reputed and renown university in Bangladesh. The ABC Convocation Management System is a cutting-edge software solution designed and implemented for ABC university to simplify and streamline the convocation ceremony processes of their institution. This system leverages the power of technology to automate convocation-related tasks, enhance coordination among various stakeholders, and ensure a seamless and memorable convocation experience for graduates and their families.

Project Proposal

Convocation ceremony is the significant part of all graduate's life where graduates are celebrated for their accomplishments and achievements. However, organizing and managing convocations can be complex and time-consuming, involving numerous logistical arrangements, attendee registration, certificate distribution, and event planning. The ABC Convocation Management System aims to alleviate these challenges and provide a comprehensive platform for convocation management. The proposed system has many features.

One of the key features of the system is attendee registration and ticketing. It enables graduates to register for the convocation ceremony online, providing a convenient and hassle-free process. Graduates can provide necessary details, select the number of guests, and generate electronic tickets for themselves and their guests. This streamlined registration process eliminates the need for manual paperwork and long queues, ensuring a smooth check-in process on the day of the event.

The system also offers efficient certificate management. It automates the generation, verification, and distribution of graduation certificates. Graduates' academic records are seamlessly integrated into the system, allowing for accurate and timely certificate generation. The certificates can be personalized and digitally signed, ensuring authenticity and eliminating the risk of manual errors. Graduates can easily collect their certificates during the convocation ceremony, saving time and reducing administrative overhead.

Moreover, the ABC Convocation Management System facilitates event planning and coordination. It enables the university administration to manage the venue selection, seating arrangements, audiovisual requirements, and other logistical aspects of the convocation ceremony. The system allows real-time communication with graduates and their families, sending automated notifications about important updates, schedules, and instructions. This ensures

that all attendees are well-informed and can plan their participation accordingly.

Additionally, the system provides data analytics and reporting functionalities. It generates comprehensive reports on attendee registration, ticket sales, certificate distribution, and other convocation-related metrics. These insights help the university administration evaluate the success of the convocation ceremony, identify areas for improvement, and make data-driven decisions for future convocations.

Class Diagram

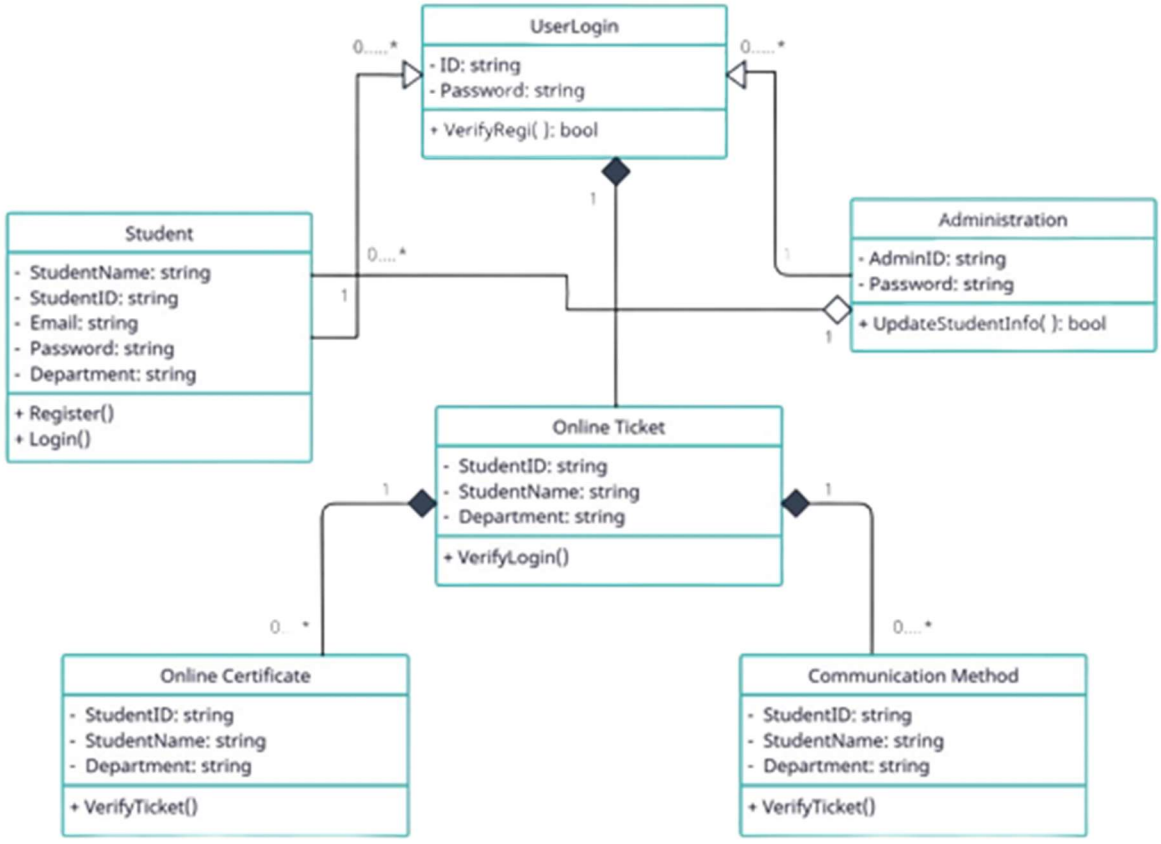


Fig: Class diagram

Use Case Diagram

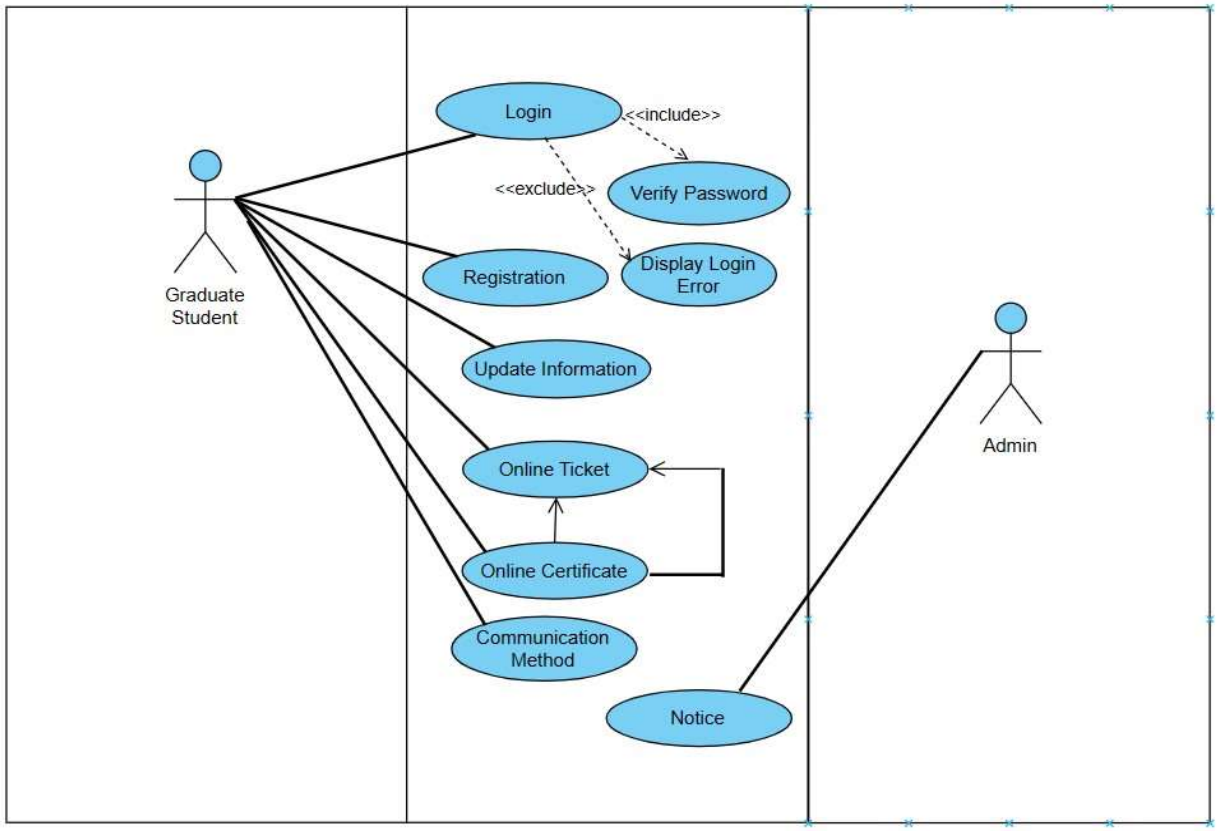


Fig: Use Case Diagram

Activity Diagram

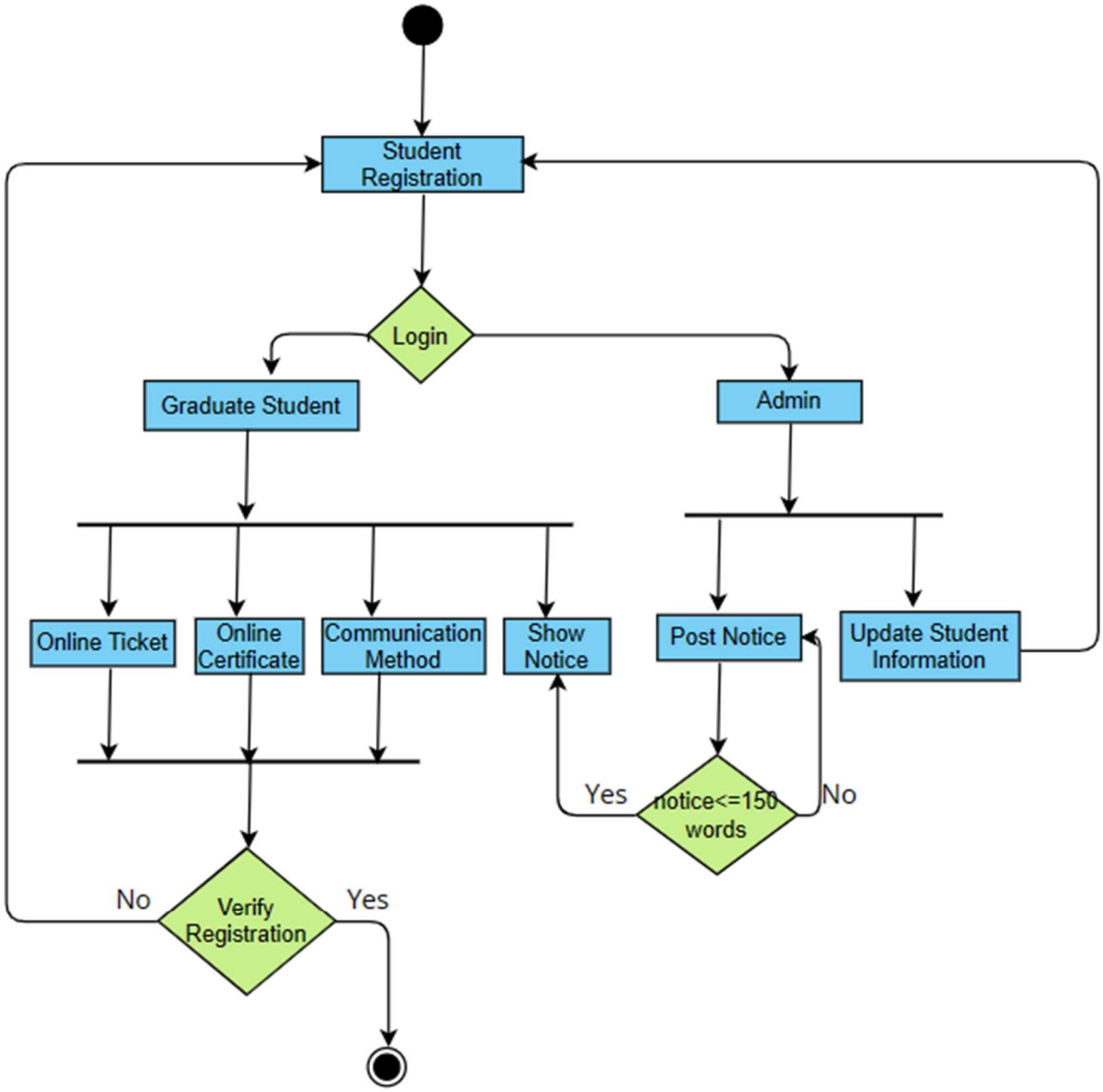
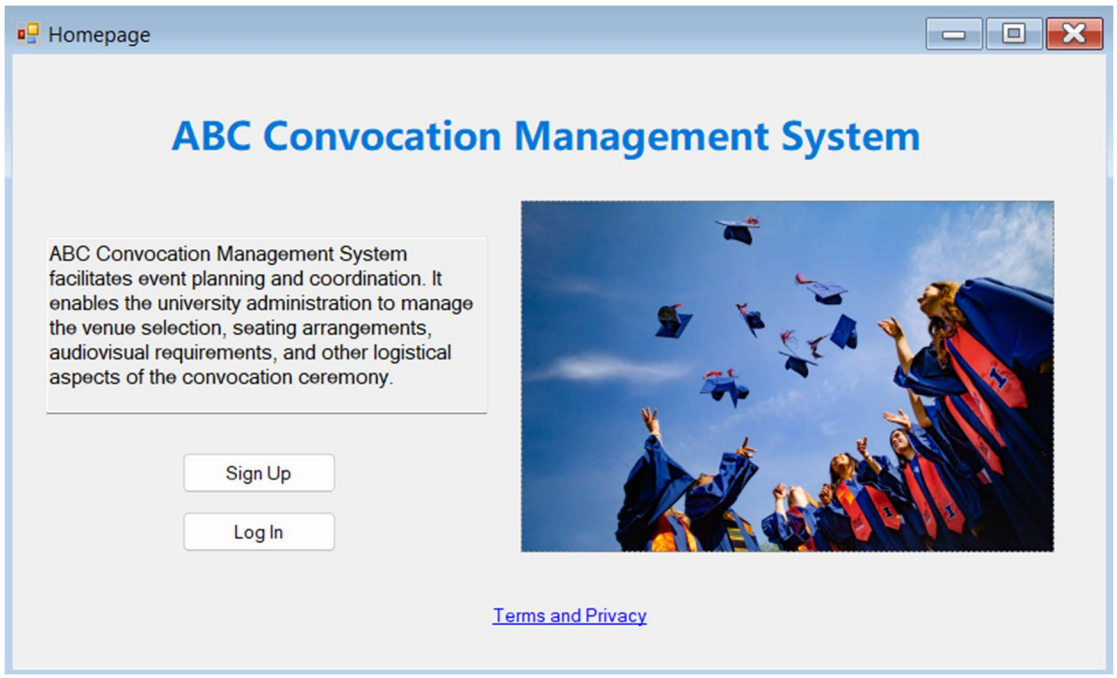
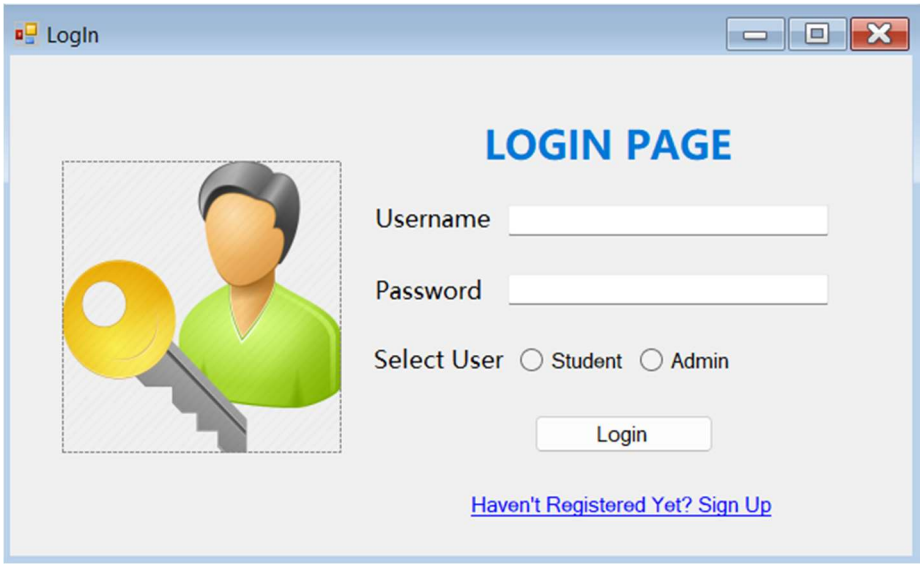
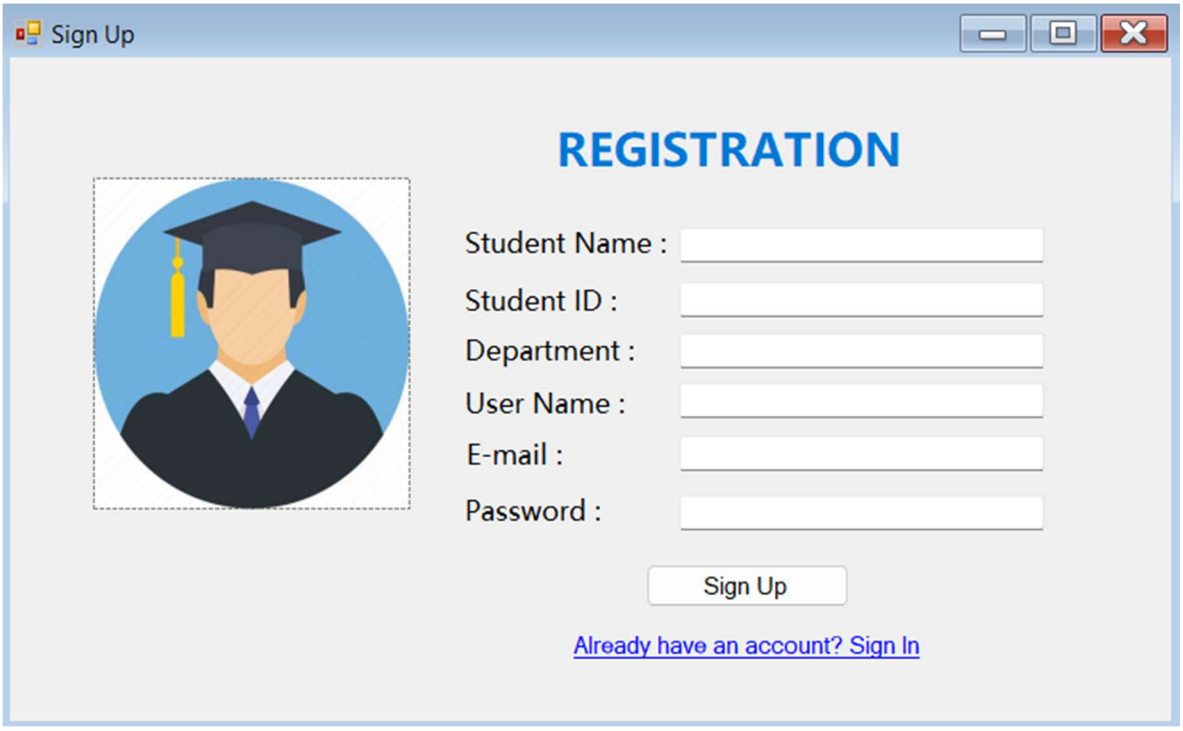


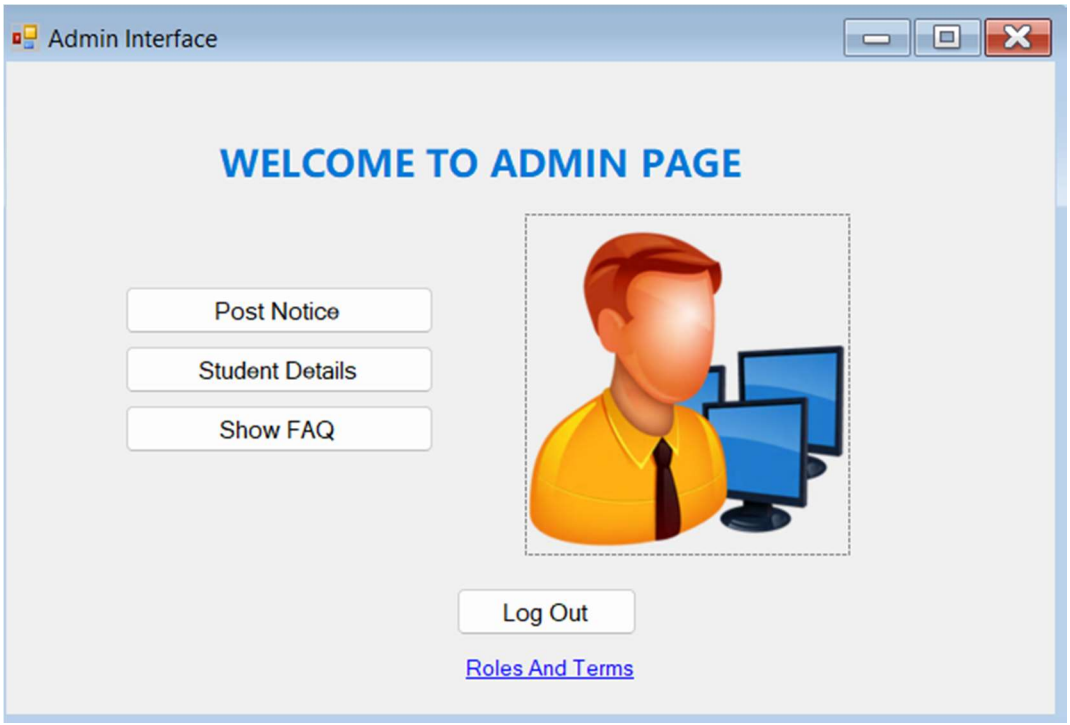
Fig: Activity Diagram

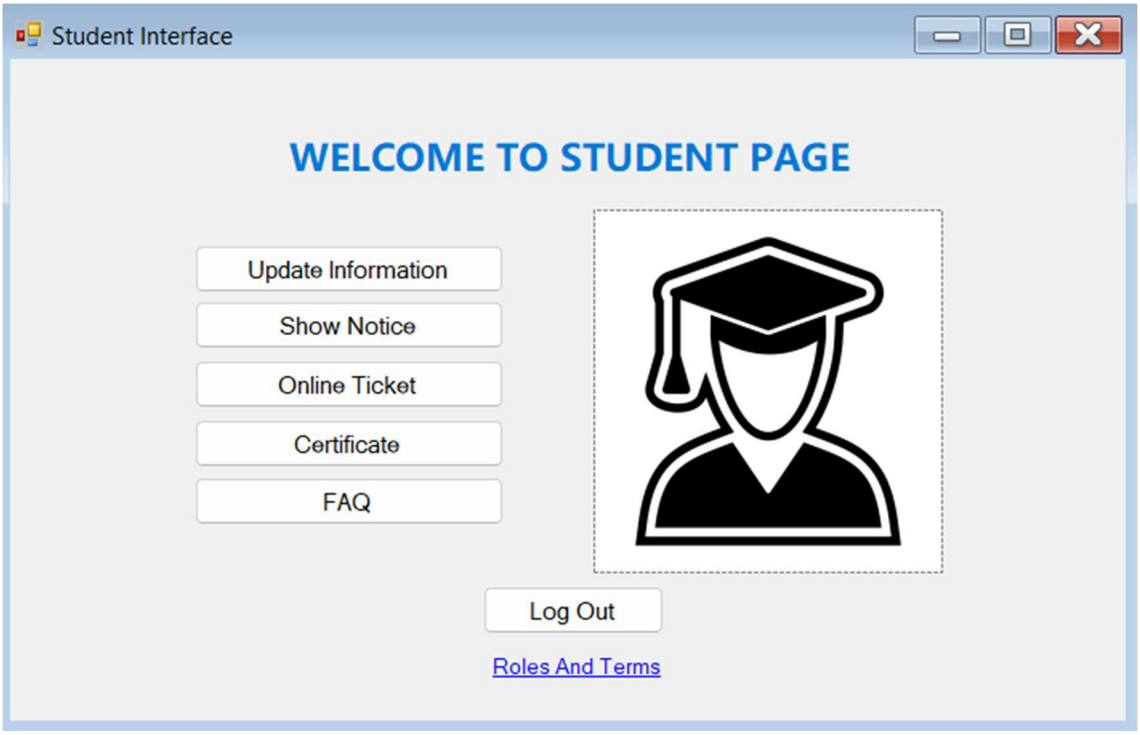
User Interface



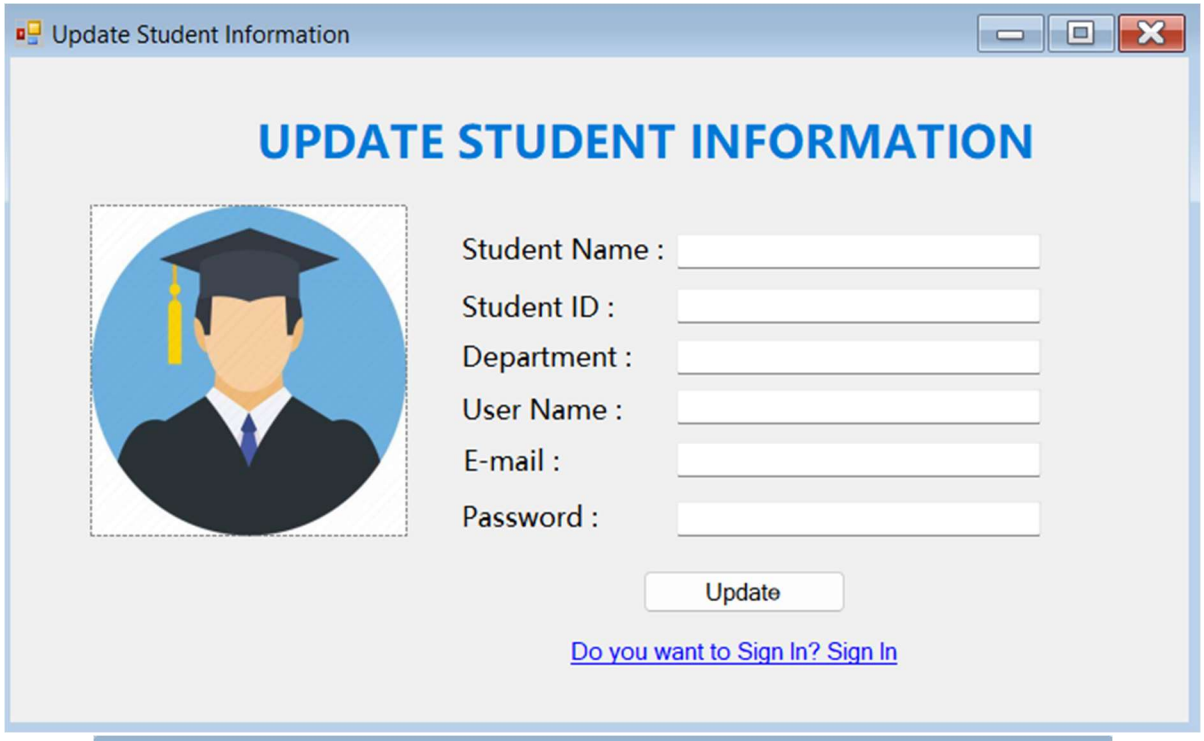


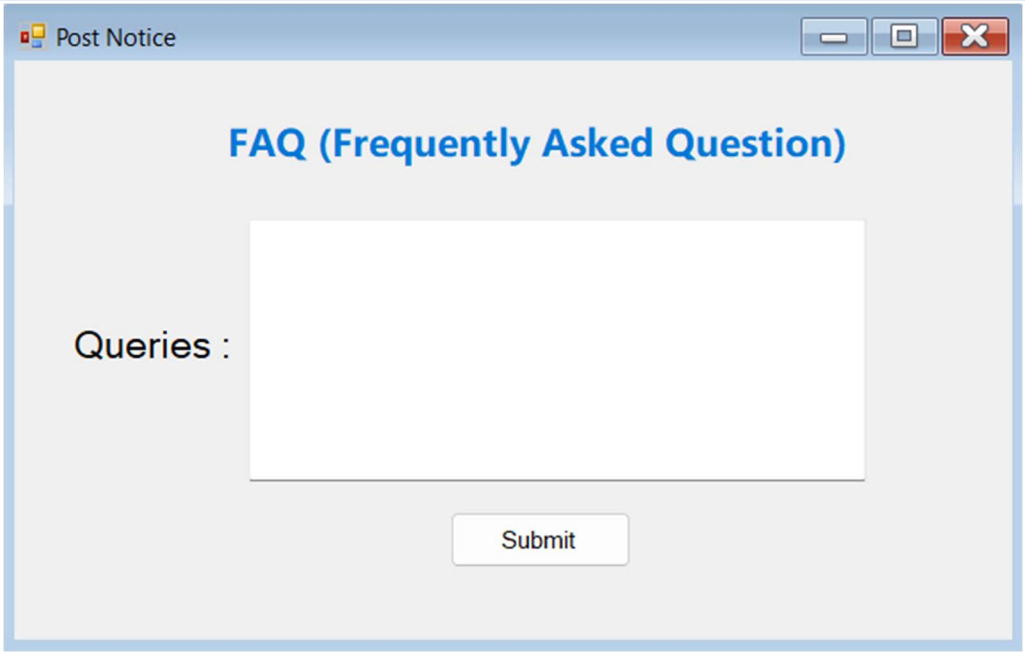

Sign Up
REGISTRATION
Student Name :
Student ID :
Department :
User Name :
E-mail :
Password :
Sign Up
Already have an account? Sign In


Admin Interface
WELCOME TO ADMIN PAGE
Post Notice
Student Details
Show FAQ
Log Out
Roles And Terms


Student Interface
WELCOME TO STUDENT PAGE
Update Information
Show Notice
Online Ticket
Certificate
FAQ
Log Out
Roles And Terms


Post Notice
POST NOTICE
Notice :
Post


Update Student Information
UPDATE STUDENT INFORMATION
Student Name :
Student ID :
Department :
User Name :
E-mail :
Password :
Update
Do you want to Sign In? Sign In


Post Notice
FAQ (Frequently Asked Question)
Queries :
Submit

Online Ticket

ONLINE TICKET PURCHASE

Student Name :

Student ID :

Department :

User Name :

E-mail :

Payment :

Purchase

Do you want to back? Student Page

Certificate

CERTIFICATE

Student Name :

Student ID :

Department :

Coupon No :

E-mail :

Collect

Do you want to back? Student Page

Scenario Description

In a convocation management system, there can be many admin and one of the admin's can login to many student's accounts, update their information, and post notices also admin already has access to many student's account. There are many graduate students who need to register first. One student can have one account. After completing the registration, the graduate students can go to login, and then they must verify their password for their own privacy. If the given password is not correct, then the graduate students will fail to login. After completing the login, they can enter and update their information in their profile. In this system, there are many registered students, that is, those who have completed their registration, only then they can apply for an online certificate through an online ticket. After reserving an online ticket, they can also communicate with the admin panel. If they are confused about any topic or if they have any queries or questions, they can send FAQ requests to admins through the features of Quires. After that, one of the admin's will check the queries and their questions and then give them answers. Admin can also update any kind of notice there to inform the graduate students. The graduate students can check the notice, which will be posted by admin.

ER Diagram

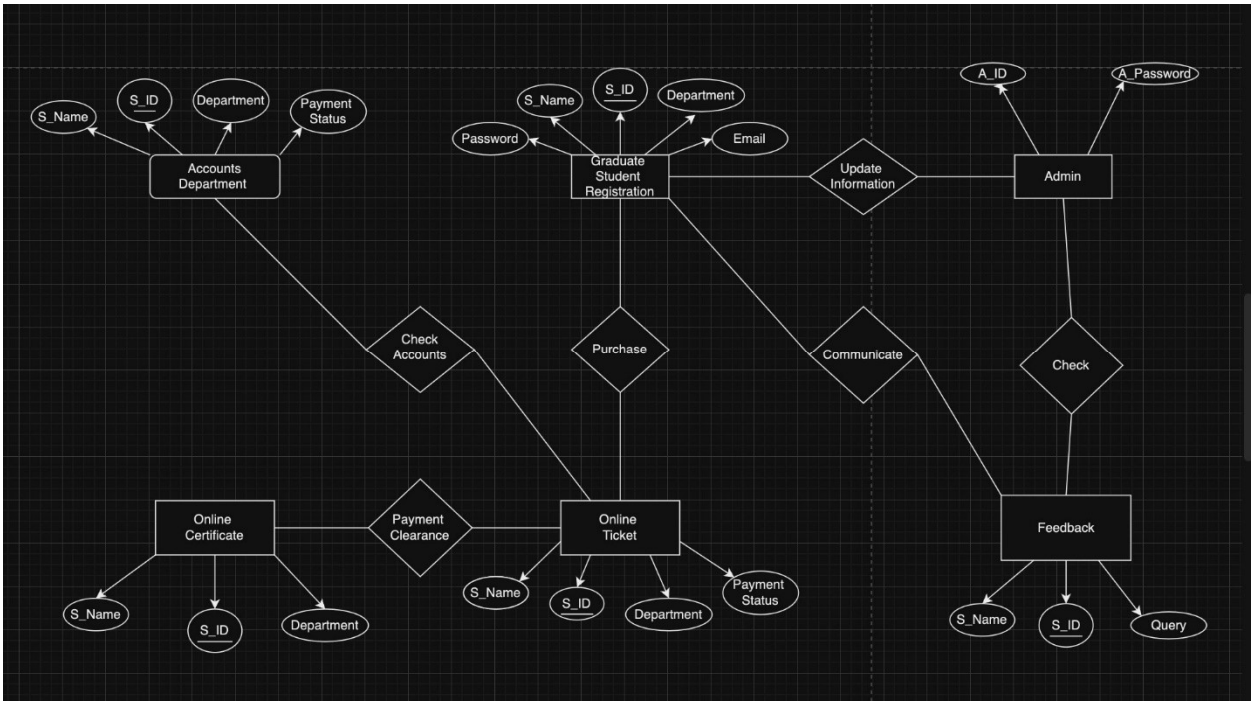


Fig: ER Diagram

Normalization

- **Check Accounts:**
  - **UNF:**
    Check Accounts (S_Name, S_ID, Department, Payment Status, S_Name, S_ID, Department, Payment Status)

  - **1NF:** There is no multi valued attribute. Relation already in 1NF.

1. S_Name, S_ID, Department, Payment Status

  - **2NF:**

1. S_Name, S_ID, Department, Payment Status

  - **3NF:** There is no transitive relationship.

1. S_Name, S_ID, Department, Payment Status

  - **Table:**

1. S_Name, S_ID, **Z_ID**
2. Z_ID, Department, Payment Status

- **Purchase:**
  - **UNF:**
    Purchase (S_Name, S_ID, Department, Payment Status, Email, Password, S_Name, S_ID, Department, Payment Status)

  - **1NF:** Email is a multivalued attribute.

1. S_Name, S_ID, Department, Payment Status, Email, Password

  - **2NF:**

1. S_Name, S_ID, Department, Payment Status
2. Email, Password

  - **3NF:**

1. S_Name, S_ID, Department, Payment Status

2. Email, Password

- **Table:**

1. S_Name, S_ID, **Z_ID**
2. Z_ID, Department, Payment Status
3. Z_ID, Email, Password

- **Communicate:**
  - **UNF:**
    Communicate (S_Name, S_ID, Department, Email, Password, Queries, S_Name, S_ID, Queries)
  - **1NF:** Email is a multivalued attribute.
    1. S_Name, S_ID, Department, Email, Password, Queries
  - **2NF:**
    1. S_Name, S_ID, Department, Email, Password
    2. S_Name, S_ID, Queries
  - **3NF:**
    1. S_Name, S_ID, Department
    2. Email, Password
    3. S_Name, S_ID, Queries
  - **Table:**
    1. S_Name, S_ID, Department, **Z_ID**
    2. Z_ID, Email, Password
    3. **Z_ID,** S_Name, S_ID, Queries
- **Update Information:**
  - **UNF:**
    Update Information (S_Name, S_ID, Department, Email, Password, A_ID, A_Password)

- **1NF:** Email is a multivalued attribute.
  1. S_Name, S_ID, Department, Email, Password, A_ID, A_Password

- **2NF:**
  1. S_Name, S_ID, Department, Email, Password
  2. A_ID, A_Password

- **3NF:**
  1. S_Name, S_ID, Department
  2. Email, Password
  3. A_ID, A_Password

- **Table:**
  1. S_Name, S_ID, Department, **P_ID**
  2. P_ID, Email, Password
  3. **P_ID**, A_ID, A_Password

- **Check:**
  - **UNF:**
    Check (S_Name, S_ID, Queries, A_ID, A_Password)

  - **1NF:** There is no multi valued attribute. Relation already in 1NF.
    1. S_Name, S_ID, Queries, A_ID, A_Password

  - **2NF:**
    1. S_Name, S_ID, Queries
    2. A_ID, A_Password

  - **3NF:** There is no transitive relationship.
    1. S_Name, S_ID, Queries
    2. A_ID, A_Password

  - **Table:**
    1. S_Name, S_ID, Queries, **P_ID**

2. **P_ID**, A_ID, A_Password

- **Payment Clearance:**
  - **UNF:**
    Payment Clearance (S_Name, S_ID, Department, S_Name, S_ID, Department, Payment Status)

  - **1NF:** There is no multi valued attribute. Relation already in 1NF.

1. S_Name, S_ID, Department, Payment Status

  - **2NF:**

1. S_Name, S_ID, Department, Payment Status

  - **3NF:** There is no transitive relationship.

1. S_Name, S_ID, Department, Payment Status

  - **Table:**

1. S_Name, S_ID, **Z_ID**
2. Z_ID, Department, Payment Status

- **Temporary Tables:**

1. S_Name, S_ID, **Z_ID**
2. Z_ID, Department, Payment Status
3. S_Name, S_ID, **Z_ID**
4. ~~Z_ID, Department, Payment Status~~
5. Z_ID, Email, Password
6. S_Name, S_ID, Department, **Z_ID**
7. ~~Z_ID, Email, Password~~
8. **Z_ID**, S_Name, S_ID, Queries
9. S_Name, S_ID, Department**, P_ID**
10. P_ID, Email, Password
11. **P_ID**, A_ID, A_Password
12. S_Name, S_ID, Queries, **P_ID**

~~13.**P_ID,** A_ID, A_Password~~
~~14.S_Name, S_ID, **Z_ID**~~
15.Z_ID, Department, Payment Status

- **Final Tables:**
  1. S_Name, S_ID, **Z_ID** ----------> Check Accounts
  2. Z_ID, Department, Payment Status --------> Check Accounts
  3. S_Name, S_ID, **Z_ID** ----------> Purchase
  4. Z_ID, Email, Password ----------> Purchase
  5. S_Name, S_ID, Department, **Z_ID** --------> Communicate
  6. **Z_ID**, S_Name, S_ID, Queries ----------> Communicate
  7. S_Name, S_ID, Department, **P_ID** --------> Update Information
  8. P_ID, Email, Password ----------> Update Information
  9. **P_ID**, A_ID, A_Password ----------> Update Information
  10. S_Name, S_ID, Queries, **P_ID** ----------> Check
  11. Z_ID, Department, Payment Status --------> Payment Clearance

Schema Diagram

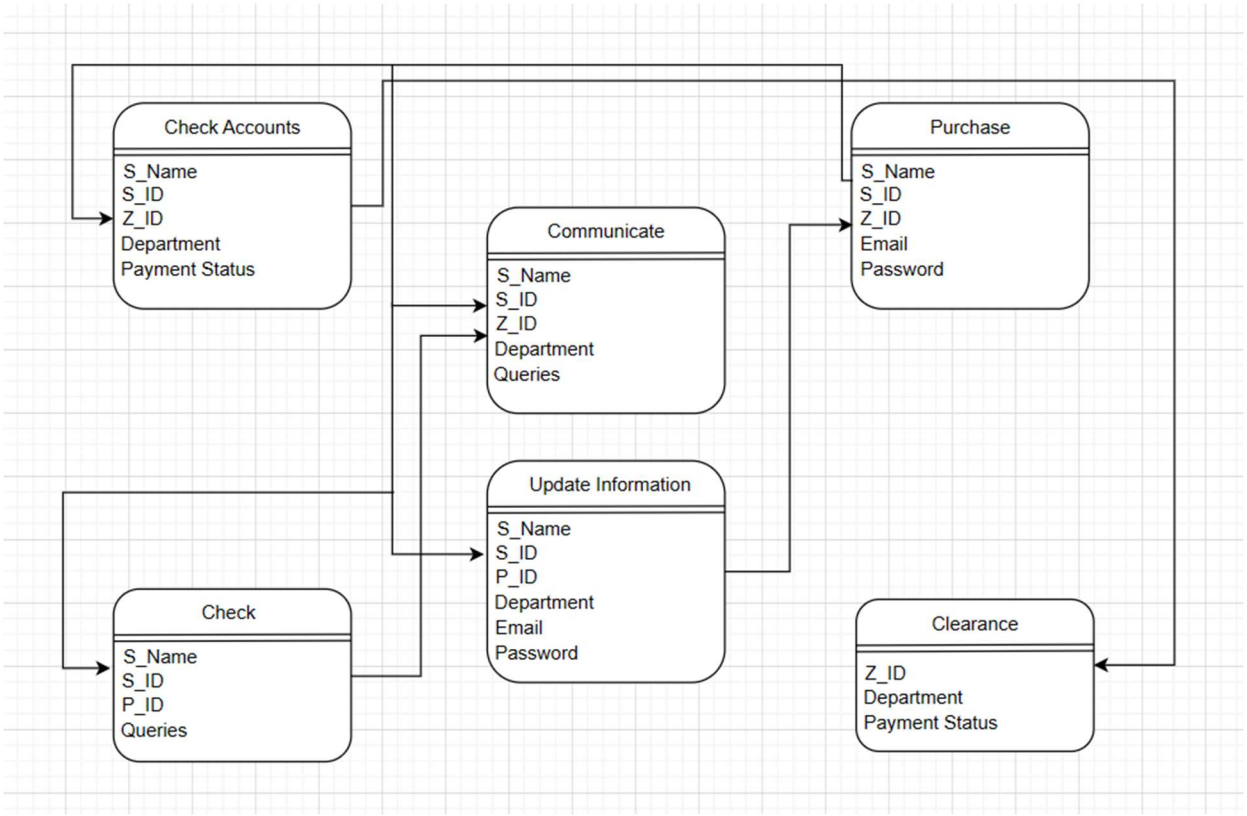


Table Creation

- **Check Accounts:**

create table checkaccounts(s_name VARCHAR2(20),s_id Number NOT NULL PRIMARY KEY,department VARCHAR2(20),paymentstatus VARCHAR2(20))

Object Type **TABLE** Object **CHECKACCOUNTS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CHECKACCOUNTS	S_NAME	Varchar2	20	-	-	-	✓	-	-
	S_ID	Number	-	-	-	1	-	-	-
	DEPARTMENT	Varchar2	20	-	-	-	✓	-	-
	PAYMENTSTATUS	Varchar2	20	-	-	-	✓	-	-
									1 - 4

- **Purchase:**

create table purchases(s_name VARCHAR2(20),s_id Number(20),email VARCHAR2(20) NOT NULL PRIMARY KEY,password VARCHAR2(20),FOREIGN KEY (s_id) REFERENCES checkaccounts(s_id))

Object Type **TABLE** Object **PURCHASES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PURCHASES	S_NAME	Varchar2	20	-	-	-	✓	-	-
	S_ID	Number	-	20	0	-	✓	-	-
	EMAIL	Varchar2	20	-	-	1	-	-	-
	PASSWORD	Varchar2	20	-	-	-	✓	-	-
									1 - 4

- **Communicate:**

create table communicate(s_name VARCHAR2(20),s_id Number(20), department VARCHAR2(20), queries VARCHAR2(20) NOT NULL PRIMARY KEY)

Object Type **TABLE** Object **COMMUNICATE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COMMUNICATE	S_NAME	Varchar2	20	-	-	-	✓	-	-
	S_ID	Number	-	20	0	-	✓	-	-
	DEPARTMENT	Varchar2	20	-	-	-	✓	-	-
	QUERIES	Varchar2	20	-	-	1	-	-	-

1 - 4

- **Update Information:**

create table updateinformation(s_name VARCHAR2(20),s_id VARCHAR2(20) NOT NULL PRIMARY KEY,department VARCHAR2(20),email VARCHAR2(20),password VARCHAR2(20),FOREIGN KEY (email) REFERENCES purchase(email))

Object Type **TABLE** Object **UPDATEINFORMATION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
UPDATEINFORMATION	S_NAME	Varchar2	20	-	-	-	✓	-	-
	S_ID	Varchar2	20	-	-	1	-	-	-
	DEPARTMENT	Varchar2	20	-	-	-	✓	-	-
	EMAIL	Varchar2	20	-	-	-	✓	-	-
	PASSWORD	Varchar2	20	-	-	-	✓	-	-

1 - 5

- **Check:**

create table checks(s_name VARCHAR2(20),s_id Number(20) NOT NULL PRIMARY KEY,queries VARCHAR2(20),FOREIGN KEY (queries) REFERENCES communicate(queries))

Object Type **TABLE** Object **CHECKS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CHECKS	S_NAME	Varchar2	20	-	-	-	✓	-	-
	S_ID	Number	-	20	0	1	-	-	-
	QUERIES	Varchar2	20	-	-	-	✓	-	-
									1 - 3

- **Clearance:**

create table clearance(department VARCHAR2(20),paymentstatus VARCHAR2(20) NOT NULL PRIMARY KEY)

Object Type **TABLE** Object **CLEARANCE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CLEARANCE	DEPARTMENT	Varchar2	20	-	-	-	✓	-	-
	PAYMENTSTATUS	Varchar2	20	-	-	1	-	-	-
									1 - 2

Sequences

- **Check Accounts:**

create sequence checkaccounts_zid INCREMENT BY 1 START WITH 1 MAXVALUE 10000000

- **Purchase:**

create sequence purchasess INCREMENT BY 1 START WITH 1 MAXVALUE 10000000

- **Communicate:**

create sequence communicates INCREMENT BY 1 START WITH 1 MAXVALUE 10000000

- **Update Information:**

create sequence update_information INCREMENT BY 1 START WITH 1 MAXVALUE 10000000

- **Check:**

create sequence checking INCREMENT BY 1 START WITH 1 MAXVALUE 10000000

- **Clearance:**

create sequence clearances INCREMENT BY 1 START WITH 1 MAXVALUE 10000000

SEQUENCE_OWNER	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
SYS	CHECKACCOUNTS_ZID	0	9999999999999999999999999	1	N	Y	10	1
SYS	PURCHASESS	1	9999999999999999999999999	1	N	Y	20	5421
SYS	COMMUNICATES	1	9999999999999999999999999	1	N	Y	20	690
SYS	UPDATE_INFORMATION	0	9999999999999999999999999	1	N	N	0	1
SYS	CHECKING	1	2000000000	1	Y	N	20	102
SYS	CLEARANCES	1	2147483647	1	N	N	20	1

Index

- **Check Accounts:**

CREATE INDEX checkaccount ON checkaccounts(s_name,s_id,department,paymentstatus);

- **Purchase:**

CREATE INDEX purchase ON purchases(s_name,s_id,email,password);

- **Communicate:**

CREATE INDEX communicates ON communicate(s_name,s_id,department,queries);

- **Update Information:**

CREATE INDEX updateinformations ON updateinformation(s_name,s_id,department,email,password);

- **Check:**

CREATE INDEX checking ON checks(s_name,s_id,queries);

- **Clearance:**

CREATE INDEX clearances ON clearance(department,paymentstatus);

Data Insertion

- **Check Accounts:**

Insert into checkaccounts(s_name,s_id,department,paymentstatus) values ('Ajoy','1','CSE','DONE');
Insert into checkaccounts(s_name,s_id,department,paymentstatus) values ('Sudipto','2','CSE','DONE');
Insert into checkaccounts(s_name,s_id,department,paymentstatus) values ('Himi','3','CSE','DONE');
Insert into checkaccounts(s_name,s_id,department,paymentstatus) values ('Mukit','4','CSE','DONE');
Insert into checkaccounts(s_name,s_id,department,paymentstatus) values ('Roy','5','CSE','DONE');

S_NAME	S_ID	DEPARTMENT	PAYMENTSTATUS
Ajoy	1	CSE	DONE
Himi	3	CSE	DONE
Mukit	4	CSE	DONE
Roy	5	CSE	DONE
Sudipto	2	CSE	DONE

- **Purchase:**

Insert into purchases(s_name,s_id,email,password) values ('Ajoy','1','ajoy@gmail.com','1987');
Insert into purchases(s_name,s_id,email,password) values ('Mukit','2','mukit@gmail.com','19gg');
Insert into purchases(s_name,s_id,email,password) values ('Himi','3','himi@gmail.com','19fd');
Insert into purchases(s_name,s_id,email,password) values ('Anika','4','anika@gmail.com','20bf');

Insert into purchases(s_name,s_id,email,password) values ('Roy','5','roiy@gmail.com','19hy');

S_NAME	S_ID	EMAIL	PASSWORD
Ajoy	1	ajoy@gmail.com	1987
Anika	4	anika@gmail.com	20bf
Himi	3	himi@gmail.com	19fd
Mukit	2	mukit@gmail.com	19gg
Roy	5	roiy@gmail.com	19hy

- **Communicate:**

Insert into communicate(s_name,s_id,department,queries) values ('Ajoy','1','CSE','givemetheupdate');

Insert into communicate(s_name,s_id,department,queries) values ('Karina','2','CSE','giveupdate');

Insert into communicate(s_name,s_id,department,queries) values ('Marina','3','IPE','showdetails');

Insert into communicate(s_name,s_id,department,queries) values ('Jorina','4','BBA','permission');

Insert into communicate(s_name,s_id,department,queries) values ('Sokina','5','EEE','showdata');

S_NAME	S_ID	DEPARTMENT	QUERIES
Ajoy	1	CSE	givemetheupdate
Jorina	4	BBA	permission
Karina	2	CSE	giveupdate
Marina	3	IPE	showdetails
Sokina	5	EEE	showdata

- **Update Information:**

Insert into updateinformation(s_name,s_id,department,email,password) values ('Ronyoy','10','CSE', ','chbd12');
Insert into updateinformation(s_name,s_id,department,email,password) values ('Ront','14','IPE', ','chbd13');
Insert into updateinformation(s_name,s_id,department,email,password) values ('Ron','17','EEE', ','chbd14');
Insert into updateinformation(s_name,s_id,department,email,password) values ('yoy','13','CSE', ','chbd15');
Insert into updateinformation(s_name,s_id,department,email,password) values ('Rony','18','BBA', ','chbd16');

S_NAME	S_ID	DEPARTMENT	EMAIL	PASSWORD
Ron	17	EEE	-	chbd14
Ront	14	IPE	-	chbd13
Rony	18	BBA	-	chbd16
Ronyoy	10	CSE	-	chbd12
yoy	13	CSE	-	chbd15

- **Check:**

Insert into checks(s_name,s_id,queries) values ('Ajoy','1','givemetheupdate');
Insert into checks(s_name,s_id,queries) values ('Roy','2','giveupdate');
Insert into checks(s_name,s_id,queries) values ('saha','3','');

Insert into checks(s_name,s_id,queries) values ('himi','6','');
Insert into checks(s_name,s_id,queries) values ('Roby','4','');

S_NAME	S_ID	QUERIES
Ajoy	1	givemetheupdate
Roby	4	-
Roy	2	giveupdate
himi	6	-
saha	3	-

- **Clearance:**

Insert into clearance(department,paymentstatus) values ('CSE','DONE');

Insert into clearance(department,paymentstatus) values ('EEE','DUE');

Insert into clearance(department,paymentstatus) values ('BBA','PartialDone');

Insert into clearance(department,paymentstatus) values ('IPE','NOTDONE');

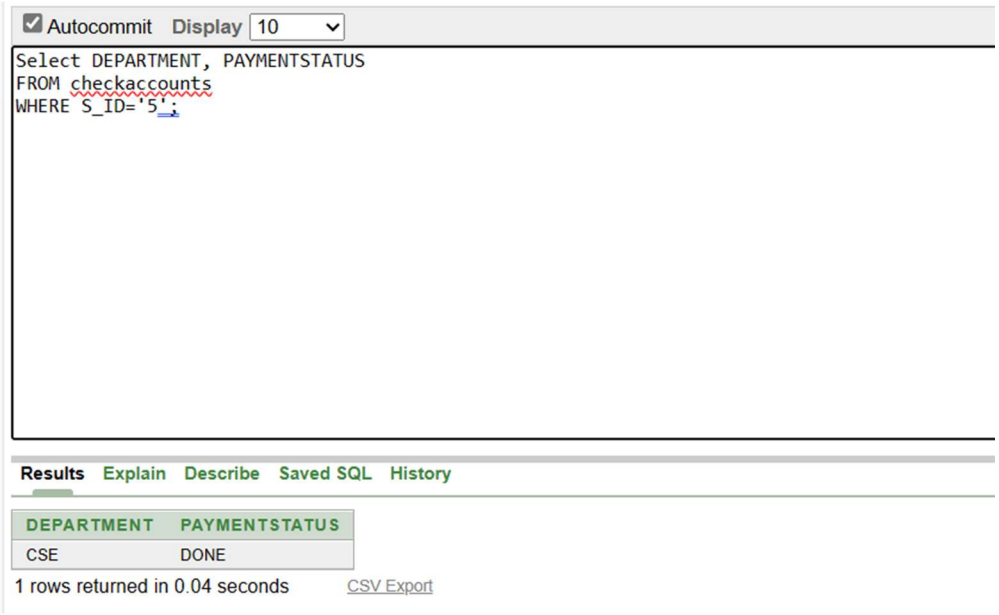
Insert into clearance(department,paymentstatus) values ('CSE','ALLDONE');

DEPARTMENT	PAYMENTSTATUS
BBA	PartialDone
CSE	ALLDONE
CSE	DONE
EEE	DUE
IPE	NOTDONE

Query Writing

❖ Single Row Function:

- Display the department and paymentstatus for s_id '5'.
  Select DEPARTMENT, PAYMENTSTATUS FROM checkaccounts WHERE S_ID='5';



- Display student id from communicate table who is from 'BBA' department.
  Select DEPARTMENT, S_id From communicate Where DEPARTMENT='BBA';

Autocommit Display 10

```
Select DEPARTMENT, S_id From communicate Where DEPARTMENT='BBA';
```

Results Explain Describe Saved SQL History

DEPARTMENT	S_ID
BBA	4

1 rows returned in 0.00 seconds CSV Export

- Display student name whose paymentstatus is 'DONE'.
  Select S_name From checkaccounts Where PAYMENTSTATUS = 'DONE';

Autocommit Display 10

```
Select S_name From checkaccounts Where PAYMENTSTATUS = 'DONE';
```

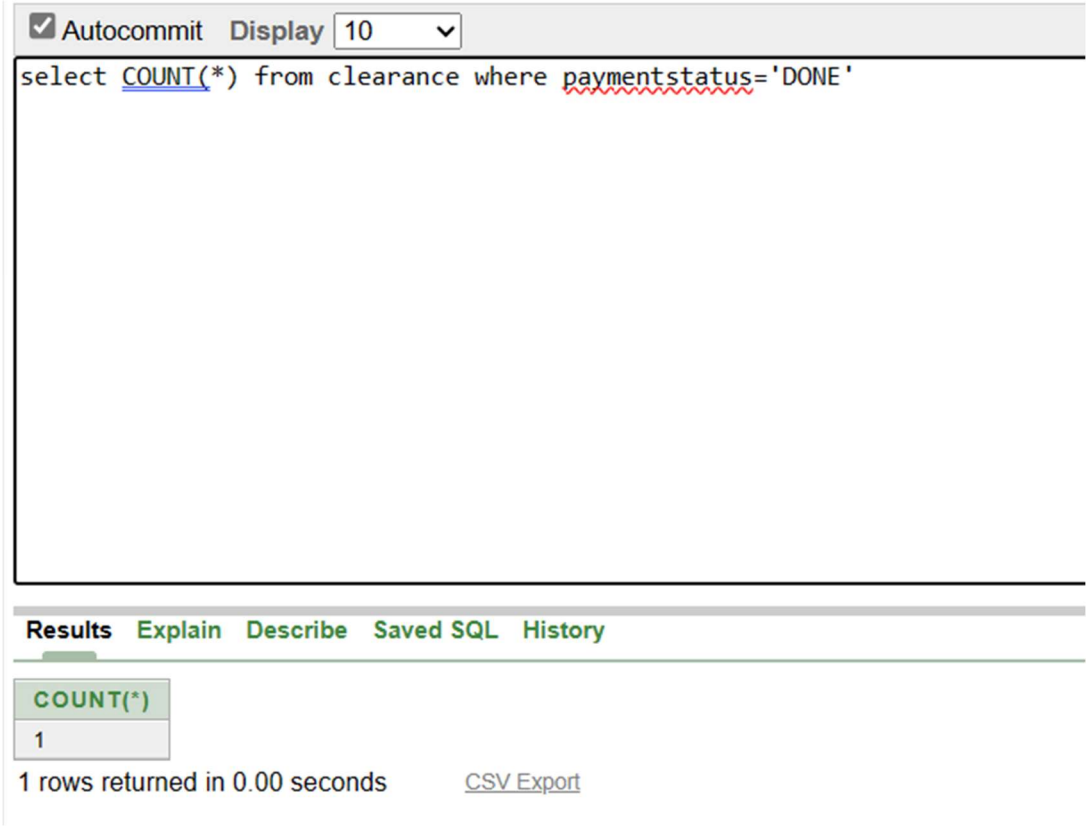
Results Explain Describe Saved SQL History

S_NAME
Ajoy
Himi
Mukit
Roy
Sudipto

5 rows returned in 0.00 seconds CSV Export

❖ Group Function:

- Show how many number of students are ready to get Online ticket.
  select COUNT(*) from clearance where paymentstatus='DONE'



- Show how many number of students have given queries.
  select COUNT(*) from checks where queries is not null

Autocommit Display 10

```
select COUNT(*) from checks where queries is not null
```

Results Explain Describe Saved SQL History

COUNT(*)
1

1 rows returned in 0.00 seconds CSV Export

- Show how many number of students are from BBA department.
select COUNT(*) from communicate where department='BBA'

Autocommit Display 10

```
select COUNT(*) from communicate where department='BBA'
```

Results Explain Describe Saved SQL History

COUNT(*)
1

1 rows returned in 0.00 seconds CSV Export

❖Subquery:

- Show the student who have registered early.
  select * from checkaccounts where s_id=(select Min(s_id) from checkaccounts)

S_NAME	S_ID	DEPARTMENT	PAYMENTSTATUS
Ajoy	1	CSE	DONE

1 rows returned in 0.03 seconds CSV Export

- Show the student details whose payment status is partially done.
  SELECT * FROM clearance WHERE paymentstatus IN (SELECT paymentstatus FROM clearance WHERE paymentstatus='PartialDone')

DEPARTMENT	PAYMENTSTATUS
BBA	PartialDone

1 rows returned in 0.00 seconds CSV Export

- Deletes the records from the clearance table for all the students whose payment is not done yet.
  DELETE FROM clearance WHERE paymentstatus IN (SELECT paymentstatus FROM clearance WHERE paymentstatus='NOTDONE')

```
1 row(s) deleted.

0.05 seconds
```

DEPARTMENT	PAYMENTSTATUS
BBA	PartialDone
CSE	ALLDONE
CSE	DONE
EEE	DUE

4 rows returned in 0.00 seconds CSV Export

❖ Joining:

- Joining the students name and students id from checkaccounts table and payment status from clearance table using EQUIJOIN as checkaccounts and clearance table has direct relation.

  SELECT checkaccounts.s_name,checkaccounts.s_id,clearance.paymentstatus FROM checkaccounts,clearance WHERE checkaccounts.department=clearance.department;

S_NAME	S_ID	PAYMENTSTATUS
Ajoy	1	ALLDONE
Ajoy	1	DONE
Himi	3	ALLDONE
Himi	3	DONE
Mukit	4	ALLDONE
Mukit	4	DONE
Roy	5	ALLDONE
Roy	5	DONE
Sudipto	2	ALLDONE
Sudipto	2	DONE

10 rows returned in 0.00 seconds CSV Export

- Joining the student name and student id from checks table and department from communicate table using EQUIJOIN as checks and communicate table has direct relation.
  SELECT checks.s_name,checks.s_id,communicate.department FROM checks,communicate WHERE checks.queries=communicate.queries;

S_NAME	S_ID	DEPARTMENT
Ajoy	1	CSE
Roy	2	CSE

2 rows returned in 0.00 seconds CSV Export

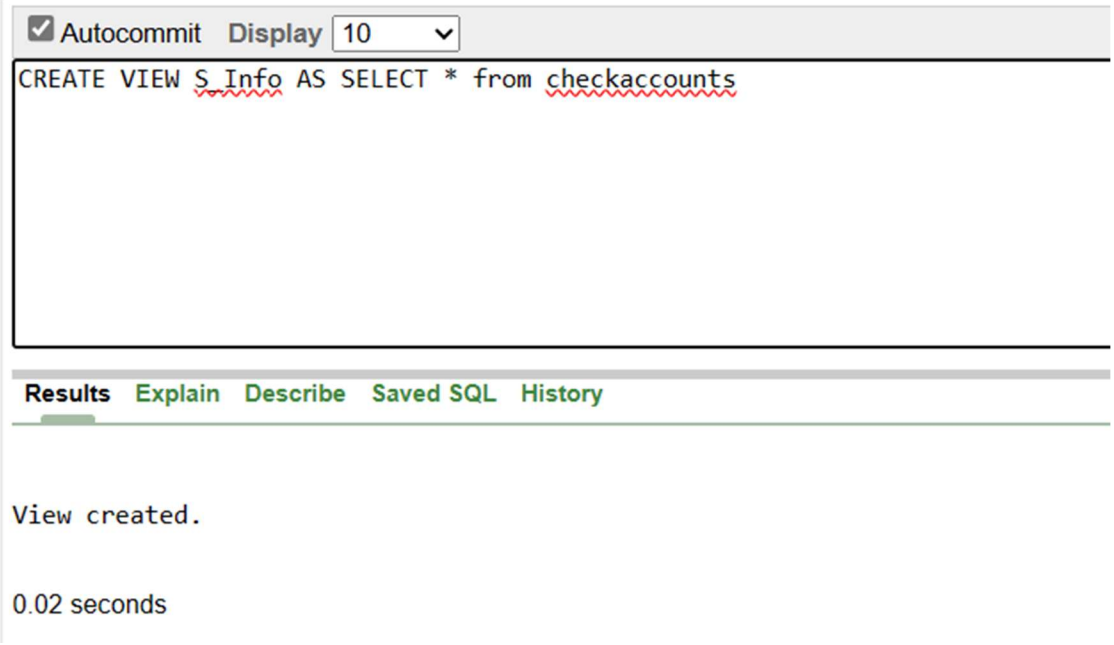
- Joining the student name and student id from checkaccounts table and department from clearance table using EQUIJOIN as checkaccounts and clearance table has direct relation.
  SELECT checkaccounts.s_name,checkaccounts.s_id,clearance.department FROM checkaccounts,clearance WHERE checkaccounts.paymentstatus=clearance.paymentstatus;

S_NAME	S_ID	DEPARTMENT
Ajoy	1	CSE
Himi	3	CSE
Mukit	4	CSE
Roy	5	CSE
Sudipto	2	CSE

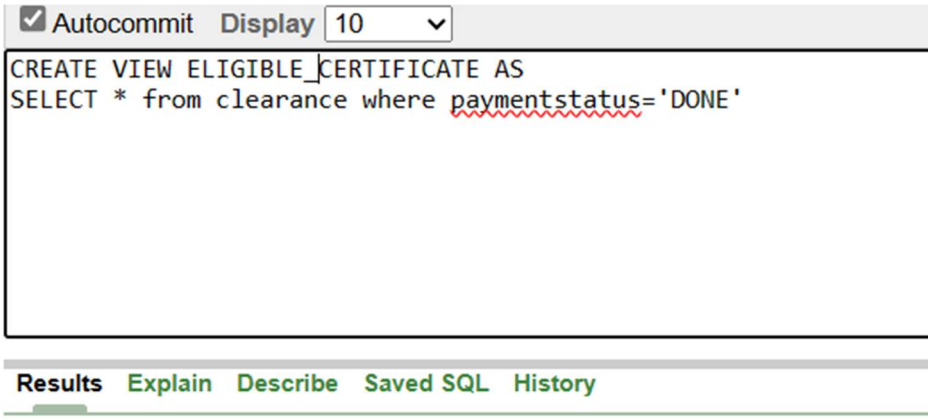
5 rows returned in 0.00 seconds CSV Export

❖ View:

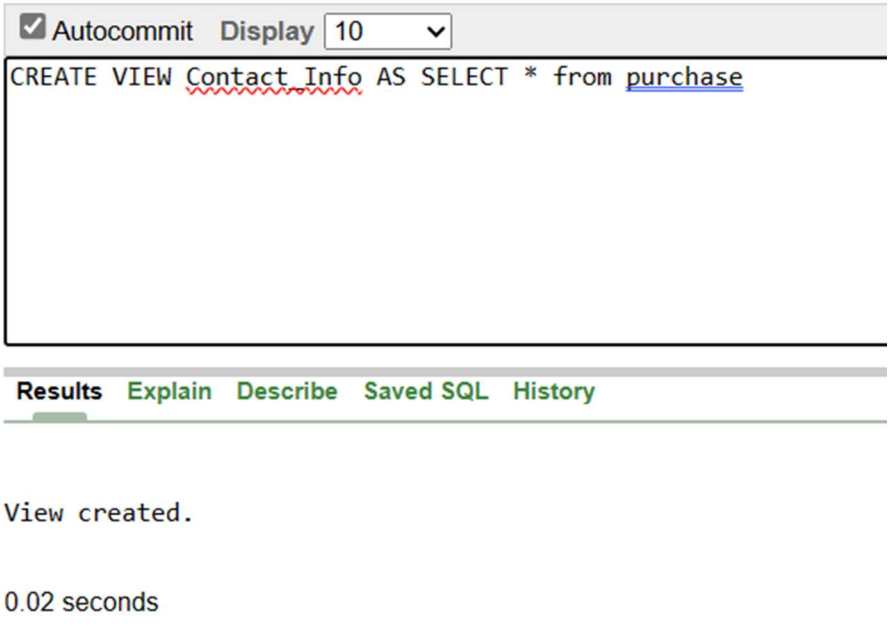
- Create a view containing all details of students
  CREATE VIEW S_Info AS SELECT * from checkaccounts



- Create a view containing all details of students who are eligible to collect online certificate
  CREATE VIEW ELIGIBLE_CERTIFICATE AS
  SELECT * from clearance where paymentstatus='DONE'

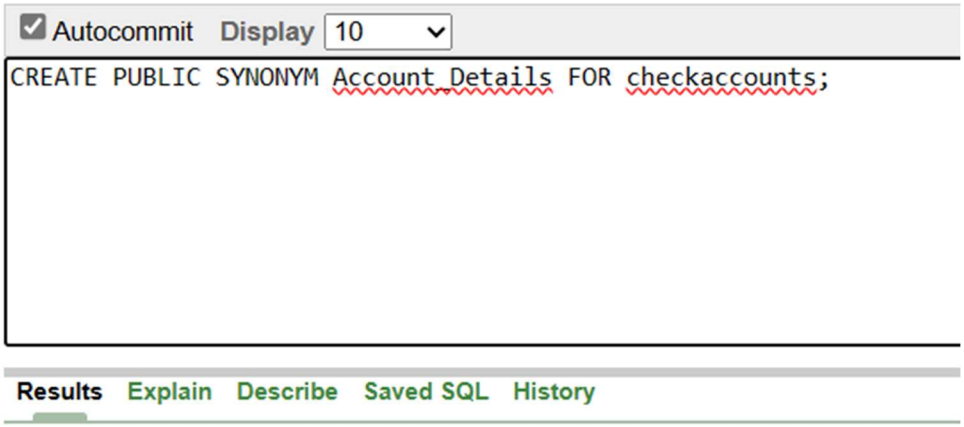


- Create a view containing all contact details of registered students.
  CREATE VIEW Contact_Info AS SELECT * from purchase

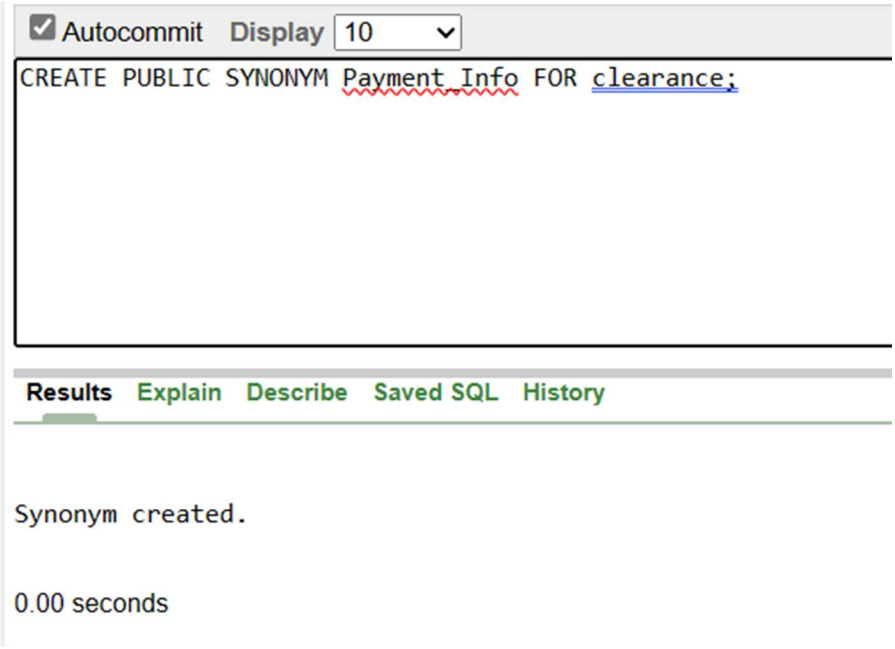


❖ Synonym:

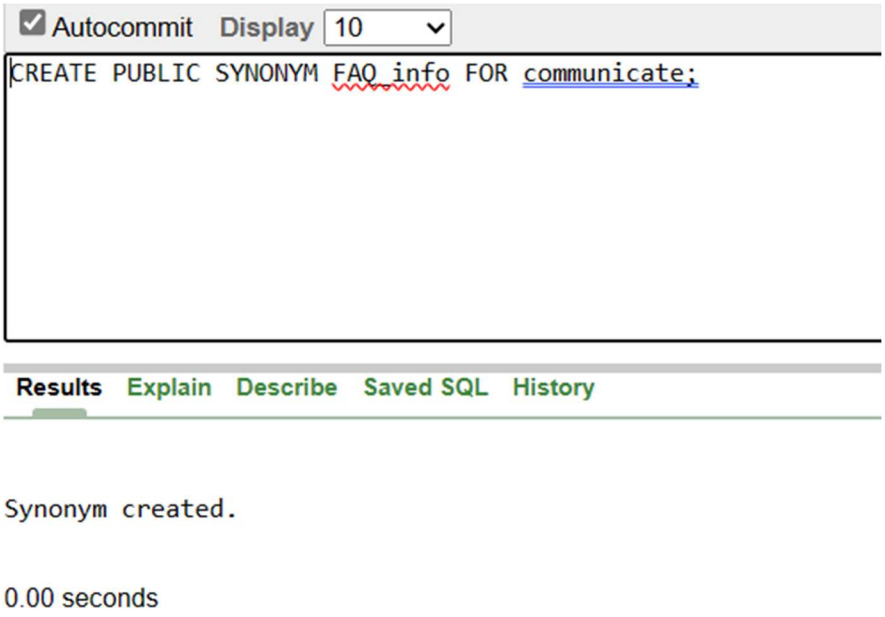
- CREATE PUBLIC SYNONYM AccountDetails FOR checkaccounts;



- CREATE PUBLIC SYNONYM Payment_Info FOR clearance;



- CREATE PUBLIC SYNONYM FAQ_info FOR communicate;



PL/SQL

❖ Function:

- Create a function to get the total count of department as “CSE” students from communicate table.

```
CREATE OR REPLACE FUNCTION get_department_count(DEPARTMENT IN varchar2)
RETURN NUMBER

IS

department_count NUMBER;

BEGIN

SELECT COUNT(*) INTO department_count

FROM communicate

WHERE DEPARTMENT = get_department_count.DEPARTMENT ;

RETURN department_count;

END;

/

DECLARE

department_count NUMBER;

BEGIN

department_count := get_department_count('CSE');

DBMS_OUTPUT.PUT_LINE('Number of students in cse department: ' || department_count);

END;
```

Autocommit Display 10

```
CREATE OR REPLACE FUNCTION get_department_count(DEPARTMENT IN varchar2) RETURN NUMBER
IS
department_count NUMBER;
BEGIN
SELECT COUNT(*) INTO department_count
FROM communicate
WHERE DEPARTMENT = get_department_count.DEPARTMENT ;
RETURN department_count;
END;
/
```

Results Explain Describe Saved SQL History

```
Function created.
```

0.04 seconds

Home > SQL > SQL Commands

Autocommit Display 10

```
DECLARE
department_count NUMBER;
BEGIN
department_count := get_department_count('CSE');
DBMS_OUTPUT.PUT_LINE('Number of students in cse department: ' || department_count);
END;
```

Results Explain Describe Saved SQL History

```
Number of students in cse department: 2

Statement processed.
```

0.02 seconds

- Create a function to get the number of students whose payment status is DONE.

```
CREATE OR REPLACE FUNCTION get_paymentstatus_count(PAYMENTSTATUS IN
varchar2) RETURN NUMBER

IS

paymentstatus_count NUMBER;

BEGIN

SELECT COUNT(*) INTO paymentstatus_count

FROM clearance

WHERE PAYMENTSTATUS = get_paymentstatus_count.PAYMENTSTATUS ;

RETURN paymentstatus_count;

END;

/

DECLARE

paymentstatus_count NUMBER;

BEGIN

paymentstatus_count := get_paymentstatus_count('DONE');

DBMS_OUTPUT.PUT_LINE('Number of students whose paymentstatus is done: ' ||
paymentstatus_count);

END;
```

Autocommit Display 10

```
CREATE OR REPLACE FUNCTION get_paymentstatus_count(PAYMENTSTATUS IN varchar2) RETURN NUMBER

IS

paymentstatus_count NUMBER;

BEGIN

SELECT COUNT(*) INTO paymentstatus_count

FROM clearance

WHERE PAYMENTSTATUS = get_paymentstatus_count.PAYMENTSTATUS ;

RETURN paymentstatus_count;

END;
```

Results Explain Describe Saved SQL History

```
Function created.
```

0.02 seconds

Autocommit Display 10

```
DECLARE

paymentstatus_count NUMBER;

BEGIN

paymentstatus_count := get_paymentstatus_count('DONE');

DBMS_OUTPUT.PUT_LINE('Number of students whose paymentstatus is done: ' || paymentstatus_count);

END;
```

Results Explain Describe Saved SQL History

```
Number of students whose paymentstatus is done: 1

Statement processed.
```

0.00 seconds

- Create a function to count the name of students whose name starts with Z.

```
CREATE OR REPLACE FUNCTION count_students(S_NAME IN varchar2) RETURN
NUMBER

IS

   total_count NUMBER;

BEGIN

   SELECT COUNT(*) INTO total_count

   FROM updateinformation WHERE S_NAME = 'Z';

   RETURN total_count;

END;

/

DECLARE

   total_count NUMBER;

BEGIN

   total_count := count_students('Z');

   DBMS_OUTPUT.PUT_LINE('Number of students whose names start with "Z": ' ||
total_count);

END;
```

Autocommit Display 10

```
CREATE OR REPLACE FUNCTION count_students(S_NAME IN varchar2) RETURN NUMBER
IS
    total_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO total_count
    FROM updateinformation WHERE S_NAME = 'Z';
    RETURN total_count;
END;
/
```

Results Explain Describe Saved SQL History

```
Function created.

0.00 seconds
```

Autocommit Display 10

```
DECLARE
    total_count NUMBER;
BEGIN
    total_count := count_students('Z');
    DBMS_OUTPUT.PUT_LINE('Number of students whose names start with ''Z'': ' || total_count);
END;
```

Results Explain Describe Saved SQL History

```
Number of students whose names start with 'Z': 0

Statement processed.

0.02 seconds
```

❖ Procedure:

- Create a procedure to update the password of the student name “Rony” from update information table.

```
CREATE OR REPLACE PROCEDURE UpdatePasswordForRony
AS
BEGIN
   UPDATE updateinformation
   SET PASSWORD = 'rony123'
   WHERE S_NAME = 'Rony';

   DBMS_OUTPUT.PUT_LINE('Password updated for Rony');
   COMMIT;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Rony not found in the database');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' ||
SQLERRM);
      ROLLBACK;
END UpdatePasswordForRony;
/
```

Autocommit Display 10

```
CREATE OR REPLACE PROCEDURE UpdatePasswordForRony AS
BEGIN
    UPDATE updateinformation
    SET PASSWORD = 'rony123'
    WHERE S_NAME = 'Rony';

    DBMS_OUTPUT.PUT_LINE('Password updated for Rony');
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Rony not found in the database');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END UpdatePasswordForRony;
/
```

Results Explain Describe Saved SQL History

```
Password updated for Rony

1 row(s) updated.
```

0.01 seconds

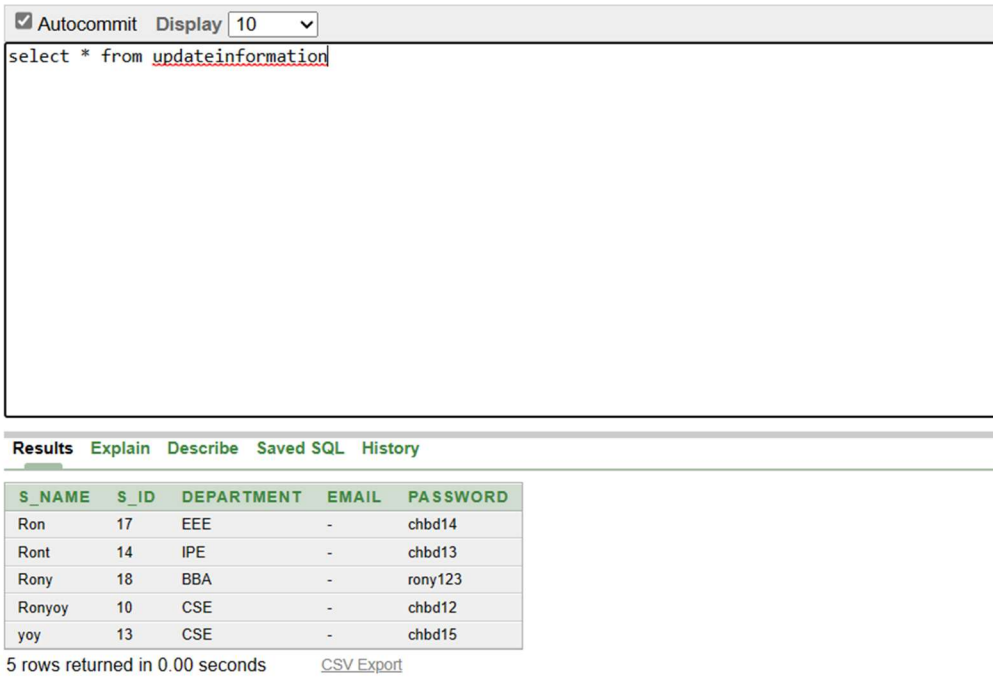
Autocommit Display 10

```
CREATE OR REPLACE PROCEDURE UpdatePasswordForRony AS
BEGIN
    UPDATE updateinformation
    SET PASSWORD = 'rony123'
    WHERE S_NAME = 'Rony';

    DBMS_OUTPUT.PUT_LINE('Password updated for Rony');
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Rony not found in the database');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK;
END UpdatePasswordForRony;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.
```

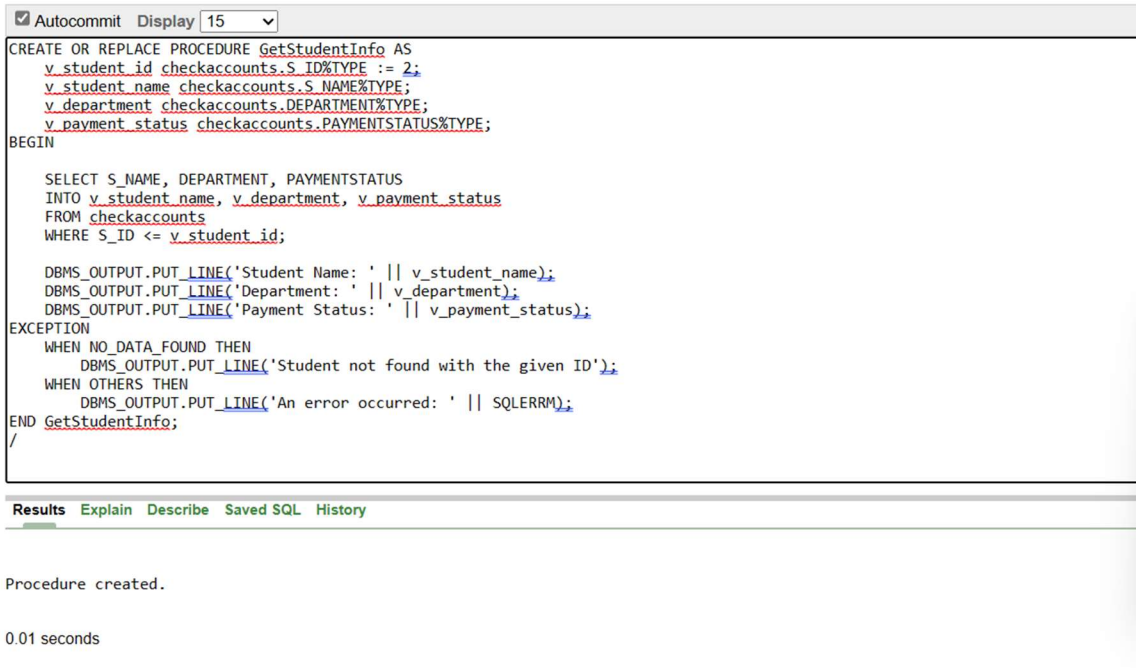
0.00 seconds



- Create a procedure to get the information about an individual student (student id, student name, payment status and department) from the checkaccounts table.

```
CREATE OR REPLACE PROCEDURE GetStudentInfo AS
   v_student_id checkaccounts.S_ID%TYPE := 2;
   v_student_name checkaccounts.S_NAME%TYPE;
   v_department checkaccounts.DEPARTMENT%TYPE;
   v_payment_status checkaccounts.PAYMENTSTATUS%TYPE;
BEGIN

   SELECT S_NAME, DEPARTMENT, PAYMENTSTATUS
   INTO v_student_name, v_department, v_payment_status
   FROM checkaccounts
   WHERE S_ID <= v_student_id;

   DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
   DBMS_OUTPUT.PUT_LINE('Department: ' || v_department);
```

```
   DBMS_OUTPUT.PUT_LINE('Payment Status: ' || v_payment_status);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Student not found with the given ID');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END GetStudentInfo;
/

BEGIN

Show GetStudentInfo;

END;

/
```



```
BEGIN
GetStudentInfo;
END;
/
```

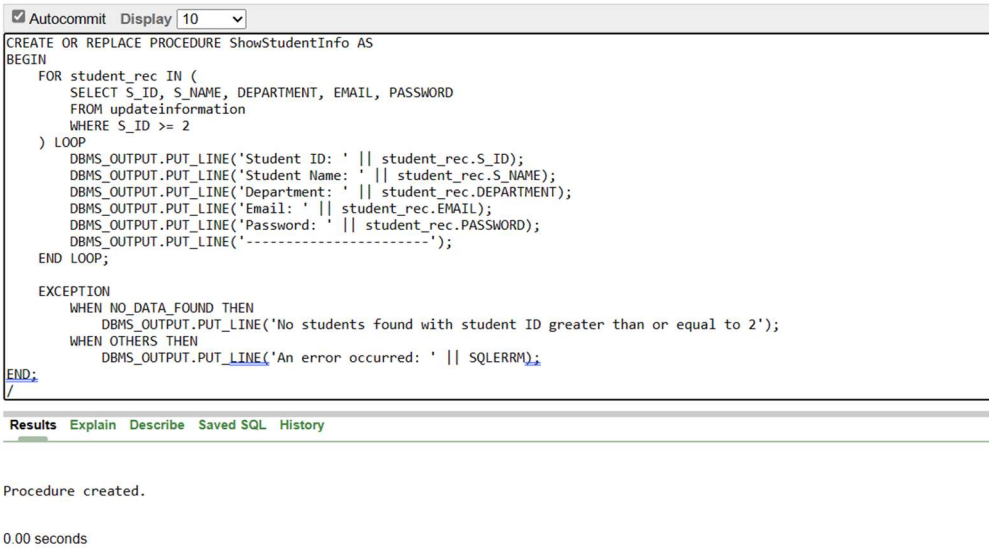
Results Explain Describe Saved SQL History

```
An error occurred: ORA-01422: exact fetch returns more than requested number of rows

Statement processed.

0.00 seconds
```

- Create a procedure to get the student information from update information table whose student id is greater than 2.

```
CREATE OR REPLACE PROCEDURE ShowStudentInfo AS
BEGIN
   FOR student_rec IN (
      SELECT S_ID, S_NAME, DEPARTMENT, EMAIL,
PASSWORD
      FROM updateinformation
      WHERE S_ID >= 2
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Student ID: ' || student_rec.S_ID);
      DBMS_OUTPUT.PUT_LINE('Student Name: ' ||
student_rec.S_NAME);
      DBMS_OUTPUT.PUT_LINE('Department: ' ||
student_rec.DEPARTMENT);
      DBMS_OUTPUT.PUT_LINE('Email: ' || student_rec.EMAIL);
      DBMS_OUTPUT.PUT_LINE('Password: ' ||
student_rec.PASSWORD);
      DBMS_OUTPUT.PUT_LINE('----------------------');
   END LOOP;

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
```

```
            DBMS_OUTPUT.PUT_LINE('No students found with student
ID greater than or equal to 2');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An error occurred: ' ||
SQLERRM);
END;
/
BEGIN
   ShowStudentInfo;
END;
/
```



Autocommit Display 10

```
CREATE OR REPLACE PROCEDURE ShowStudentInfo AS
BEGIN
    FOR student_rec IN (
        SELECT S_ID, S_NAME, DEPARTMENT, EMAIL, PASSWORD
        FROM updateinformation
        WHERE S_ID >= 2
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Student ID: ' || student_rec.S_ID);
        DBMS_OUTPUT.PUT_LINE('Student Name: ' || student_rec.S_NAME);
        DBMS_OUTPUT.PUT_LINE('Department: ' || student_rec.DEPARTMENT);
        DBMS_OUTPUT.PUT_LINE('Email: ' || student_rec.EMAIL);
        DBMS_OUTPUT.PUT_LINE('Password: ' || student_rec.PASSWORD);
        DBMS_OUTPUT.PUT_LINE('----------------------');
    END LOOP;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No students found with student ID greater than or equal to 2');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.

0.00 seconds
```

```
BEGIN
    ShowStudentInfo;
END;
/
```

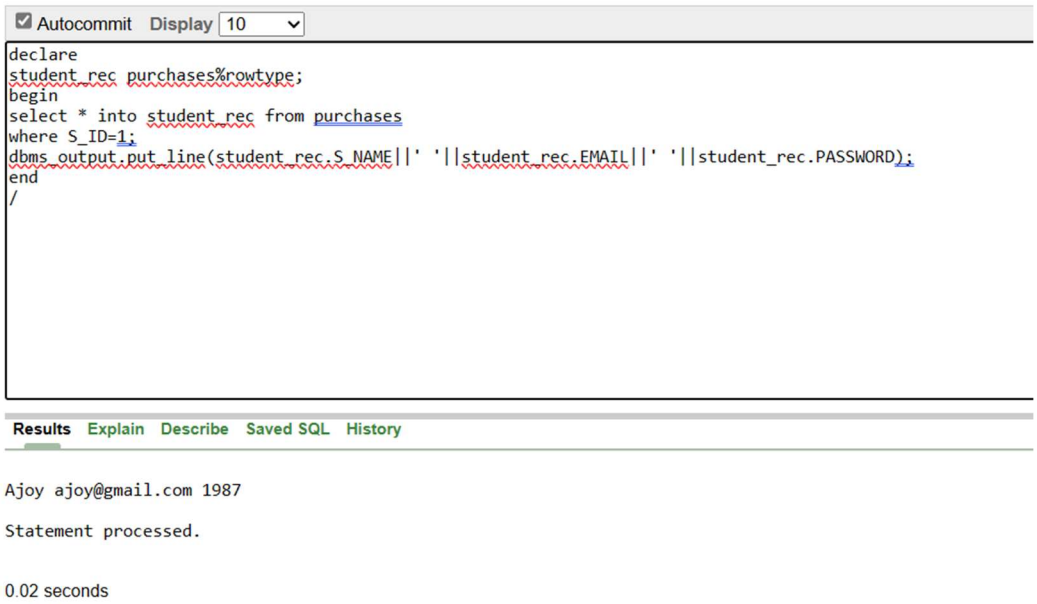
Results Explain Describe Saved SQL History

```
Student ID: 17
Student Name: Ron
Department: EEE
Email:
Password: chbd14
-----------------------
Student ID: 14
Student Name: Ront
Department: IPE
Email:
Password: chbd13
-----------------------
Student ID: 18
Student Name: Rony
Department: BBA
Email:
Password: rony123
-----------------------
Student ID: 10
Student Name: Ronyoy
Department: CSE
Email:
Password: chbd12
-----------------------
Student ID: 13
Student Name: yoy
Department: CSE
Email:
Password: chbd15
-----------------------

Statement processed.

0.00 seconds
```

❖ Record:

- Create a record for purchase table. The record should return student name, email and password of a student on the basis of student id.

```
declare
student_rec purchases%rowtype;
begin
select * into student_rec from purchases
where S_ID=1;
dbms_output.put_line(student_rec.S_NAME||' '||student_rec.EMAIL||'
'||student_rec.PASSWORD);
end
/
```



- Create a record for communicate table. The record should return student id and student name on the basis of department.

```
declare
cursor c_department is
select * from communicate where DEPARTMENT='CSE';
rec_department communicate%rowtype;
begin
open c_department;
fetch c_department into rec_department;
dbms_output.put_line(rec_department.S_ID||'
'||rec_department.S_NAME);
close c_department;
end;
/
```

Autocommit Display 10

```
declare
cursor c_department is
select * from communicate where DEPARTMENT='CSE';
rec_department communicate%rowtype;
begin
open c_department;
fetch c_department into rec_department;
dbms_output.put_line(rec_department.S_ID||' '||rec_department.S_NAME);
close c_department;
end;
/
```

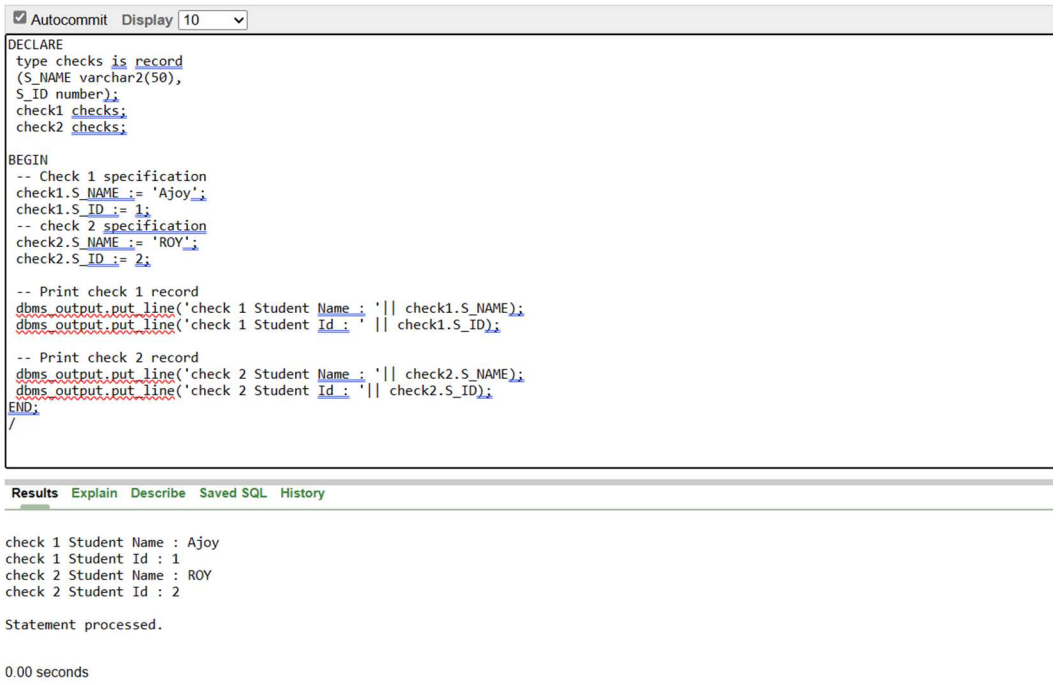
Results Explain Describe Saved SQL History

```
1 Ajoy

Statement processed.

0.00 seconds
```

- Create a user defined record to enter and return student name and student id of atleast two student.

```
DECLARE
 type checks is record
 (S_NAME varchar2(50),
 S_ID number);
 check1 checks;
 check2 checks;

BEGIN
 -- Check 1 specification
 check1.S_NAME := 'Ajoy';
 check1.S_ID := 1;
 -- check 2 specification
 check2.S_NAME := 'ROY';
 check2.S_ID := 2;
```

```
-- Print check 1 record
dbms_output.put_line('check 1 Student Name : '|| check1.S_NAME);
dbms_output.put_line('check 1 Student Id : ' || check1.S_ID);

-- Print check 2 record
dbms_output.put_line('check 2 Student Name : '|| check2.S_NAME);
dbms_output.put_line('check 2 Student Id : '|| check2.S_ID);
END;
/
```

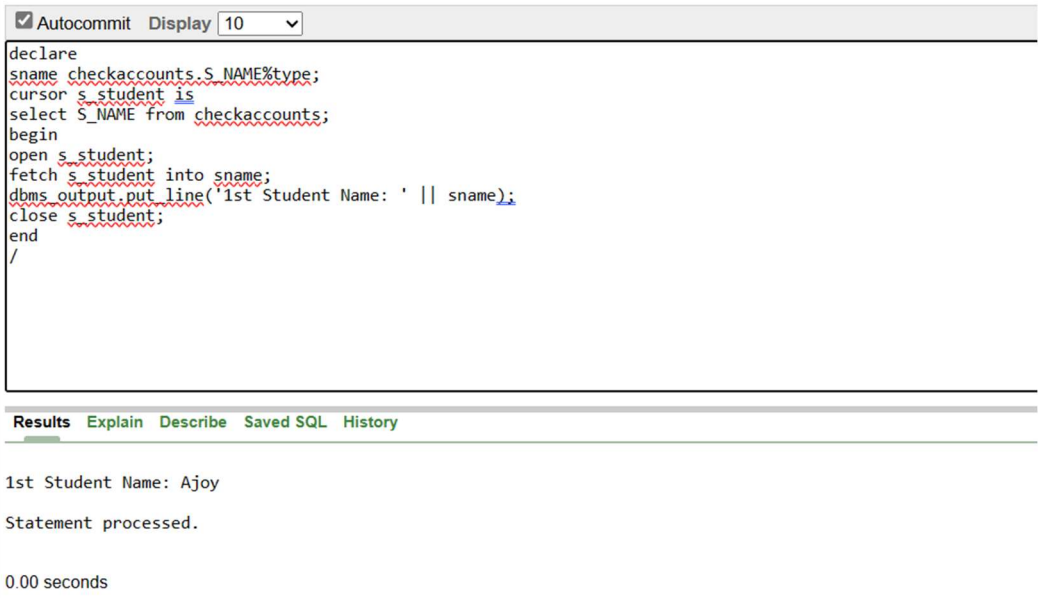


❖ Cursor :

- Create a cursor that can output 1st Student's name in checkaccounts table.

```
declare
sname checkaccounts.S_NAME%type;
cursor s_student is
```

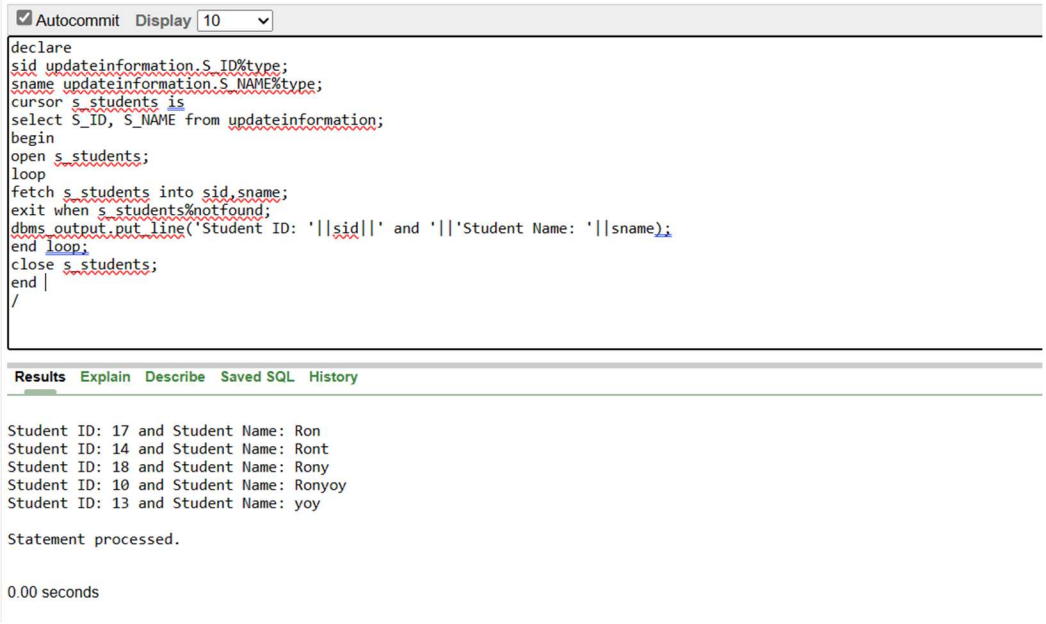
```
select S_NAME from checkaccounts;
begin
open s_student;
fetch s_student into sname;
dbms_output.put_line('1st Student Name: ' || sname);
close s_student;
end
/
```



- Create a cursor that can output the id and name of all the students in updateinformation table.

```
declare
sid updateinformation.S_ID%type;
sname updateinformation.S_NAME%type;
cursor s_students is
select S_ID, S_NAME from updateinformation;
begin
open s_students;
loop
fetch s_students into sid,sname;
```

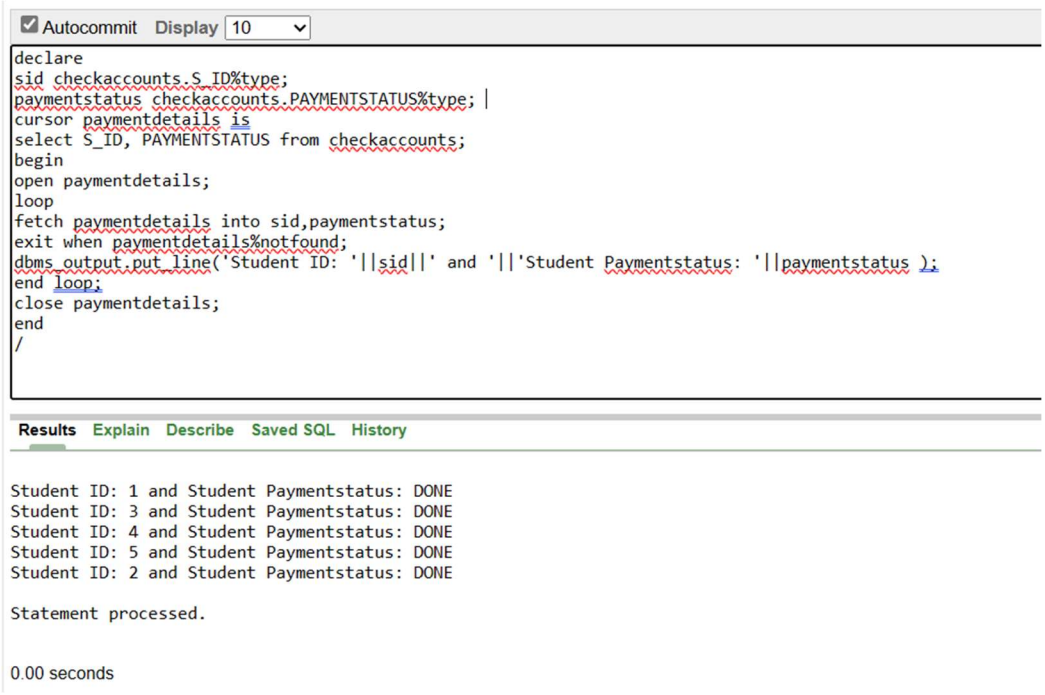
```
exit when s_students%notfound;
dbms_output.put_line('Student ID: '||sid||' and '||'Student Name:
'||sname);
end loop;
close s_students;
end
/
```



- Create a cursor that can output the id and payment status of all the students.

```
declare
sid checkaccounts.S_ID%type;
paymentstatus checkaccounts.PAYMENTSTATUS%type;
cursor paymentdetails is
select S_ID, PAYMENTSTATUS from checkaccounts;
begin
open paymentdetails;
loop
fetch paymentdetails into sid,paymentstatus;
```

```
exit when paymentdetails%notfound;
dbms_output.put_line('Student ID: '||sid||' and '||'Student
Paymentstatus: '||paymentstatus );
end loop;
close paymentdetails;
end
/
```

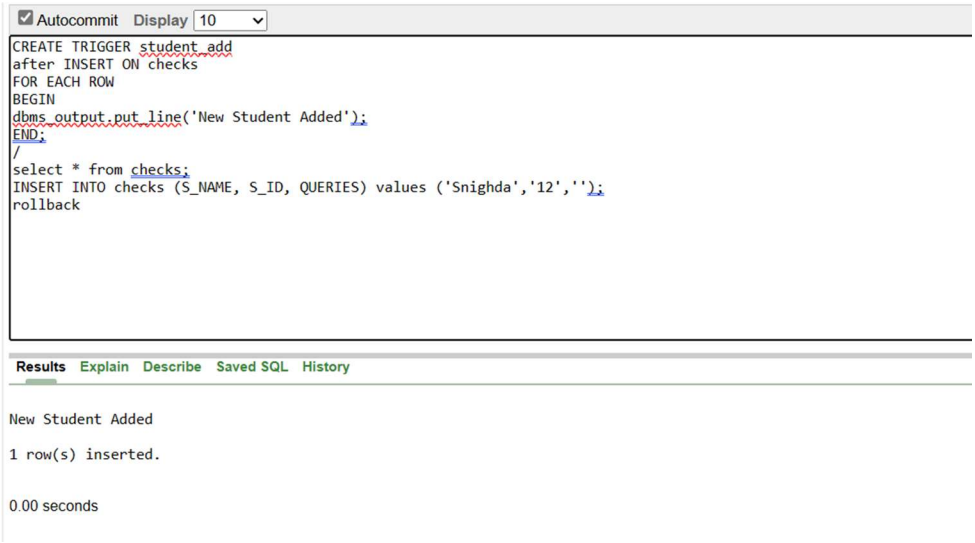


❖ Trigger:

- Create a trigger in such a way that whenever a new row is inserted into the checks table an output ‘New Student Added’ is generated.

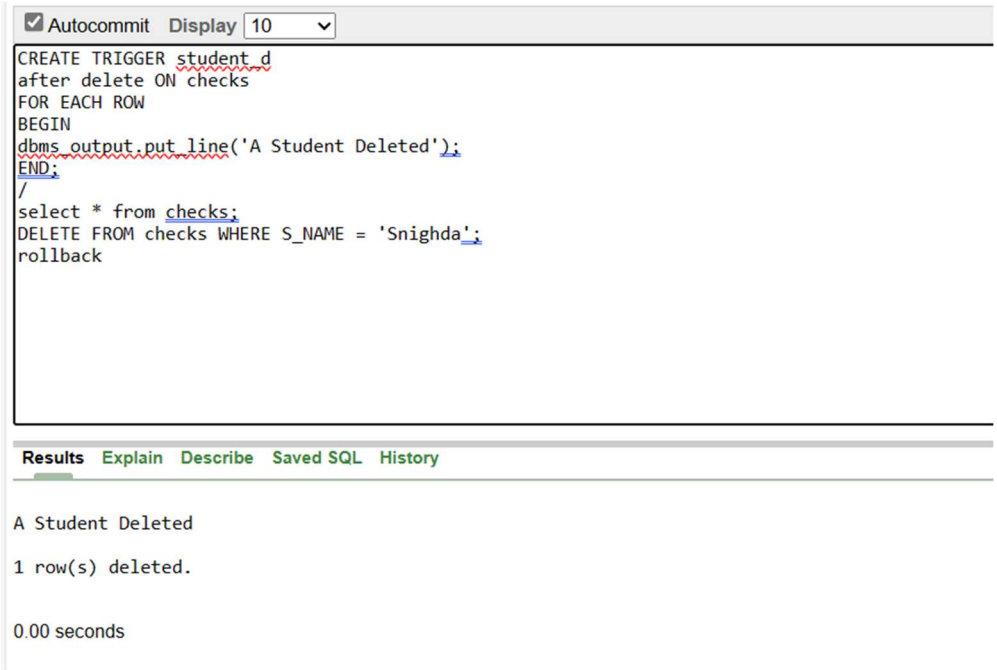
```
CREATE TRIGGER student_add
after INSERT ON checks
FOR EACH ROW
BEGIN
dbms_output.put_line('New Student Added');
```

```
END;
/
select * from checks;
INSERT INTO checks (S_NAME, S_ID, QUERIES) values
('Snighda','12','');
Rollback
```



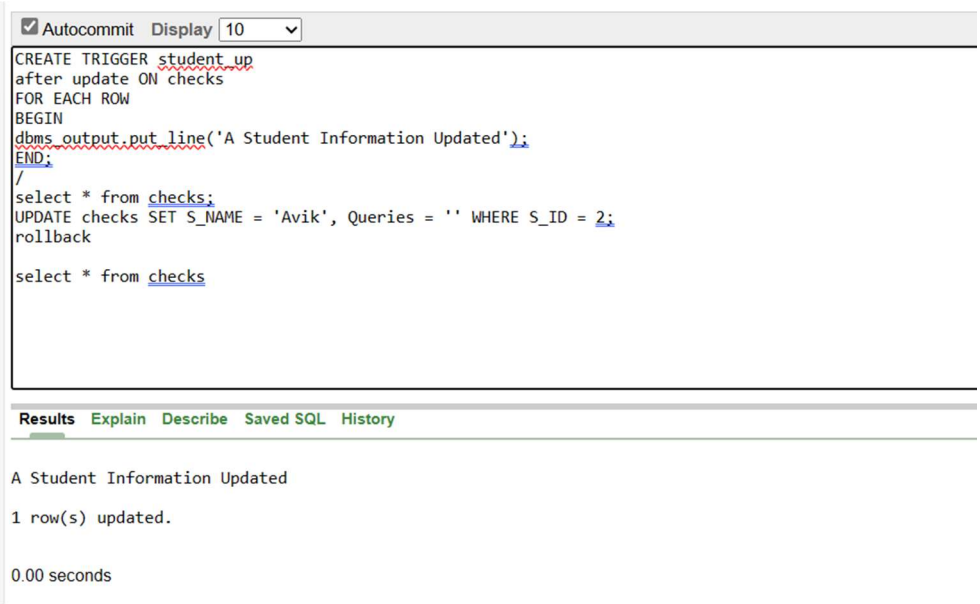
- Create a trigger in such a way that whenever a row is deleted from the checks table an output ‘A Student Deleted’ is generated.

```
CREATE TRIGGER student_d
after delete ON checks
FOR EACH ROW
BEGIN
dbms_output.put_line('A Student Deleted');
END;
/
select * from checks;
DELETE FROM checks WHERE S_NAME = 'Snighda';
rollback
```



- Create a trigger in such a way that whenever a row is updated from the checks table an output ‘A Student Information Updated’ is generated.

```
CREATE TRIGGER student_up
after update ON checks
FOR EACH ROW
BEGIN
dbms_output.put_line('A Student Information Updated');
END;
/
select * from checks;
UPDATE checks SET S_NAME = 'Avik', Queries = '' WHERE S_ID
= 2;
rollback

select * from checks
```



Autocommit Display 10

```
CREATE TRIGGER student_up
after update ON checks
FOR EACH ROW
BEGIN
dbms_output.put_line('A Student Information Updated');
END;
/
select * from checks;
UPDATE checks SET S_NAME = 'Avik', Queries = '' WHERE S_ID = 2;
rollback

select * from checks
```

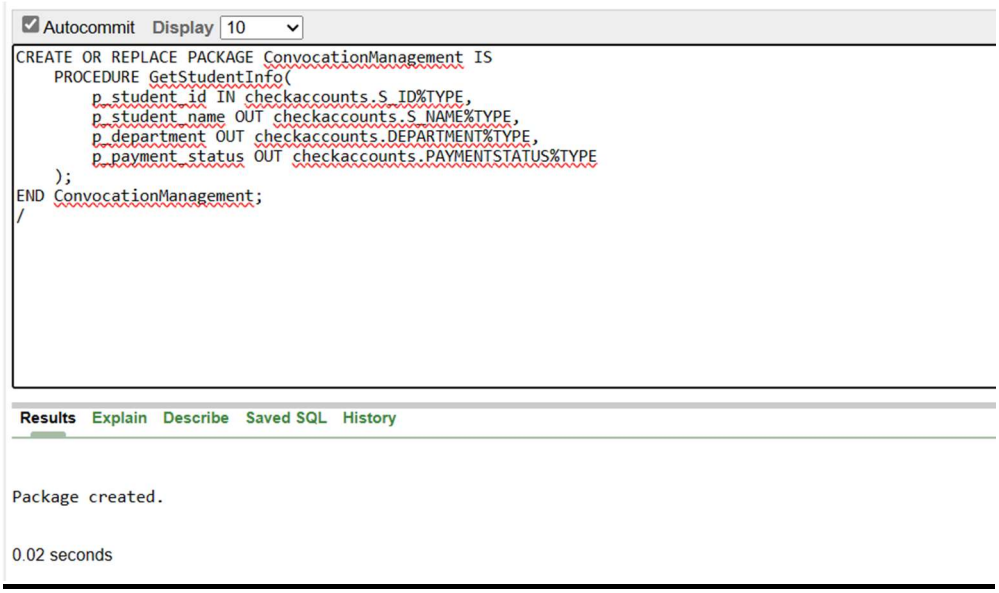
Results Explain Describe Saved SQL History

```
A Student Information Updated

1 row(s) updated.


0.00 seconds
```

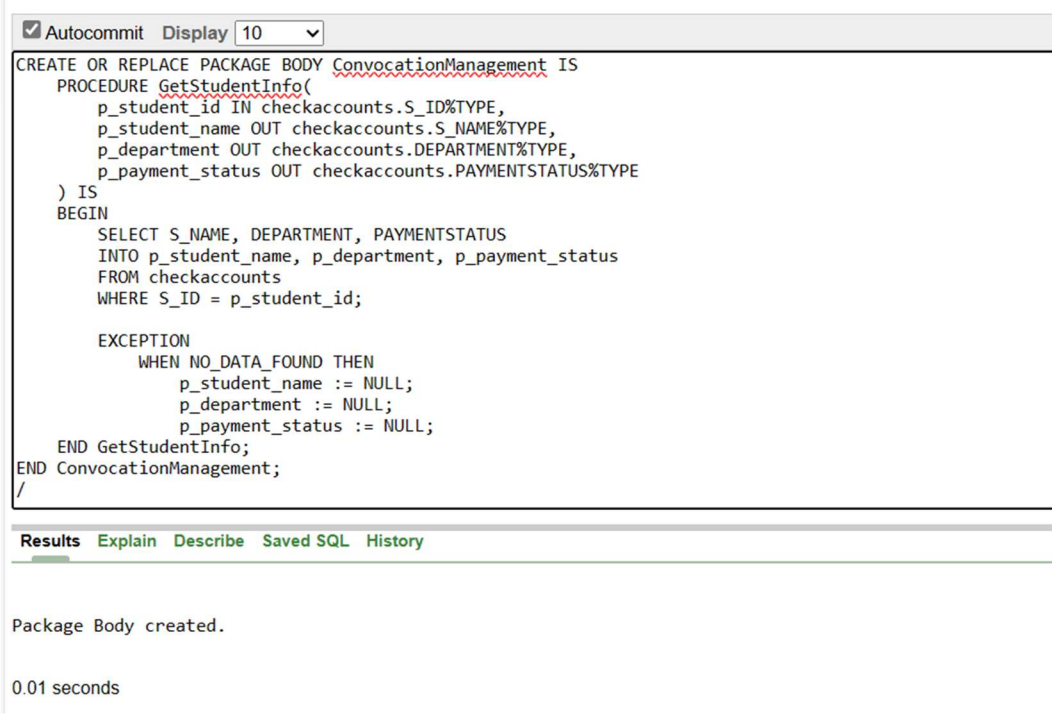
❖ Package :

- Create a package on checkaccounts table to return student name, payment status and department by student id ;

```
CREATE OR REPLACE PACKAGE ConvocationManagement IS
   PROCEDURE GetStudentInfo(
      p_student_id IN checkaccounts.S_ID%TYPE,
      p_student_name OUT checkaccounts.S_NAME%TYPE,
      p_department OUT checkaccounts.DEPARTMENT%TYPE,
      p_payment_status OUT
checkaccounts.PAYMENTSTATUS%TYPE
   );
END ConvocationManagement;
/
```




```
CREATE OR REPLACE PACKAGE BODY ConvocationManagement IS
   PROCEDURE GetStudentInfo(
      p_student_id IN checkaccounts.S_ID%TYPE,
      p_student_name OUT checkaccounts.S_NAME%TYPE,
      p_department OUT checkaccounts.DEPARTMENT%TYPE,
      p_payment_status OUT checkaccounts.PAYMENTSTATUS%TYPE
   ) IS
   BEGIN
      SELECT S_NAME, DEPARTMENT, PAYMENTSTATUS
      INTO p_student_name, p_department, p_payment_status
      FROM checkaccounts
      WHERE S_ID = p_student_id;

      EXCEPTION
         WHEN NO_DATA_FOUND THEN
            p_student_name := NULL;
            p_department := NULL;
            p_payment_status := NULL;
   END GetStudentInfo;
END ConvocationManagement;
/
```



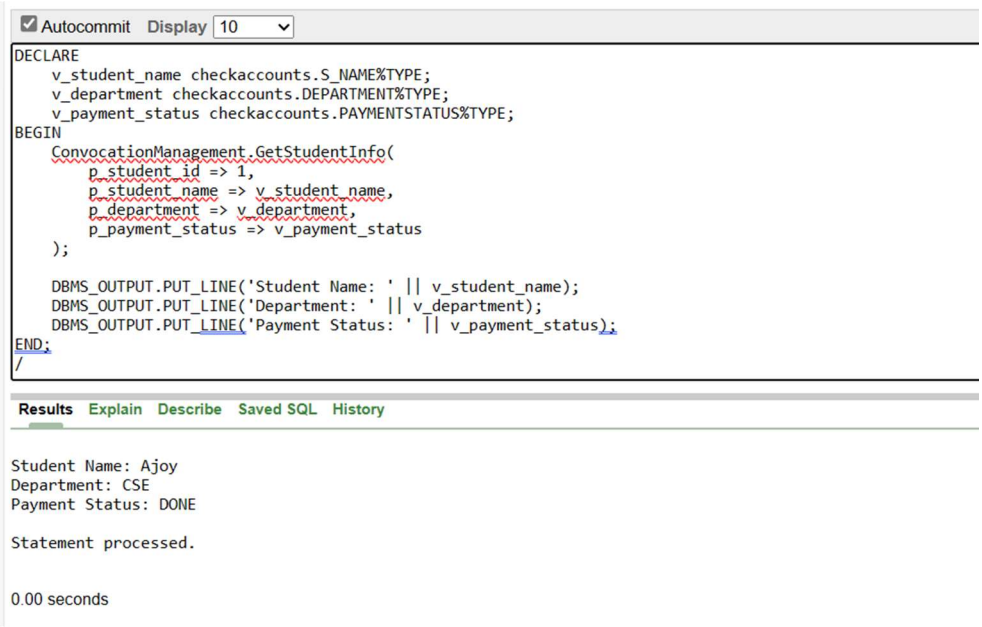
Autocommit Display 10

```
CREATE OR REPLACE PACKAGE BODY ConvocationManagement IS
    PROCEDURE GetStudentInfo(
        p_student_id IN checkaccounts.S_ID%TYPE,
        p_student_name OUT checkaccounts.S_NAME%TYPE,
        p_department OUT checkaccounts.DEPARTMENT%TYPE,
        p_payment_status OUT checkaccounts.PAYMENTSTATUS%TYPE
    ) IS
    BEGIN
        SELECT S_NAME, DEPARTMENT, PAYMENTSTATUS
        INTO p_student_name, p_department, p_payment_status
        FROM checkaccounts
        WHERE S_ID = p_student_id;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                p_student_name := NULL;
                p_department := NULL;
                p_payment_status := NULL;
    END GetStudentInfo;
END ConvocationManagement;
/
```

Results Explain Describe Saved SQL History

```
Package Body created.

0.01 seconds
```

```
DECLARE
   v_student_name checkaccounts.S_NAME%TYPE;
   v_department checkaccounts.DEPARTMENT%TYPE;
   v_payment_status checkaccounts.PAYMENTSTATUS%TYPE;
BEGIN
   ConvocationManagement.GetStudentInfo(
      p_student_id => 1,
      p_student_name => v_student_name,
      p_department => v_department,
      p_payment_status => v_payment_status
   );

   DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
   DBMS_OUTPUT.PUT_LINE('Department: ' || v_department);
   DBMS_OUTPUT.PUT_LINE('Payment Status: ' ||
v_payment_status);
END;
/
```



Autocommit Display 10

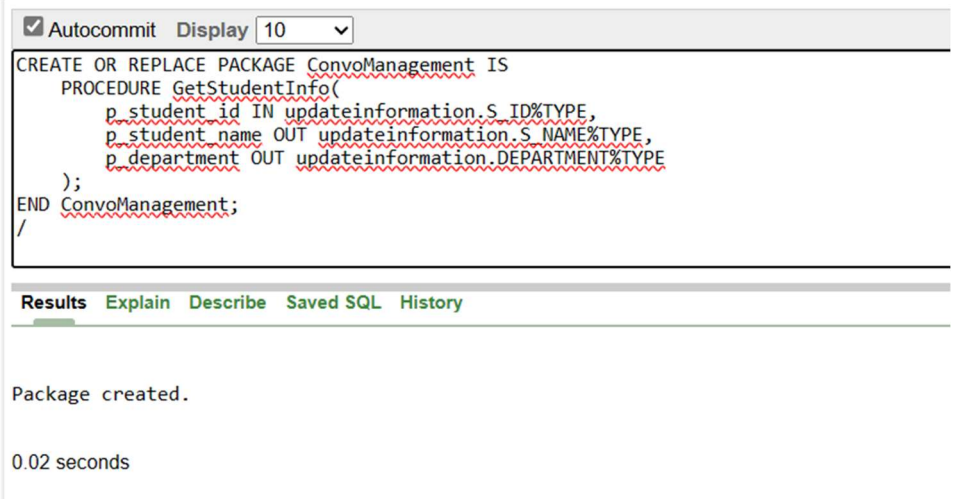
```
DECLARE
    v_student_name checkaccounts.S_NAME%TYPE;
    v_department checkaccounts.DEPARTMENT%TYPE;
    v_payment_status checkaccounts.PAYMENTSTATUS%TYPE;
BEGIN
    ConvocationManagement.GetStudentInfo(
        p_student_id => 1,
        p_student_name => v_student_name,
        p_department => v_department,
        p_payment_status => v_payment_status
    );

    DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
    DBMS_OUTPUT.PUT_LINE('Department: ' || v_department);
    DBMS_OUTPUT.PUT_LINE('Payment Status: ' || v_payment_status);
END;
/
```

Results Explain Describe Saved SQL History

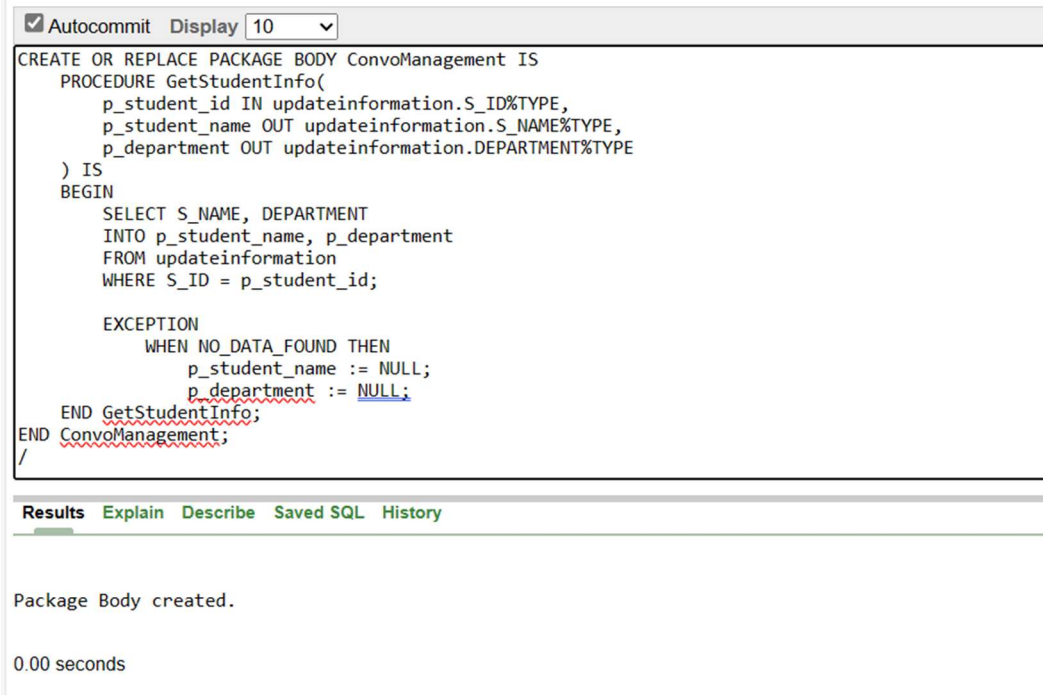
```
Student Name: Ajoy
Department: CSE
Payment Status: DONE

Statement processed.


0.00 seconds
```

- Create a package on updateinformation table to return student name and department by student id ;

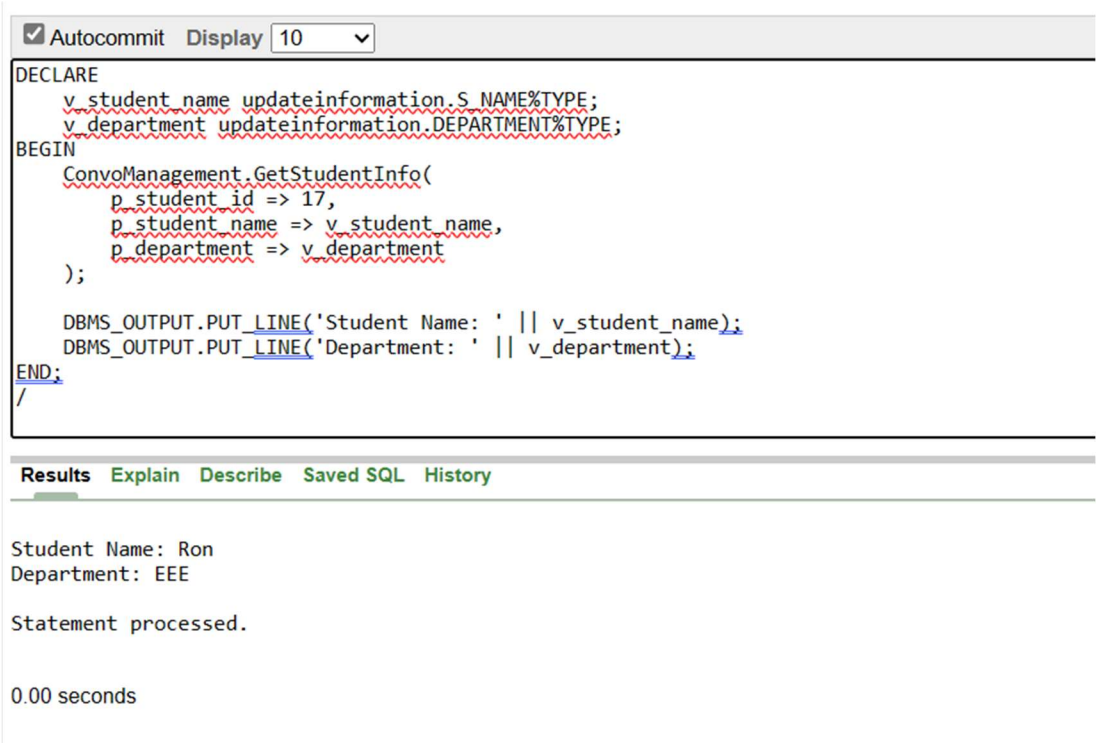
```
CREATE OR REPLACE PACKAGE ConvoManagement IS
   PROCEDURE GetStudentInfo(
      p_student_id IN updateinformation.S_ID%TYPE,
      p_student_name OUT updateinformation.S_NAME%TYPE,
      p_department OUT updateinformation.DEPARTMENT%TYPE
   );
END ConvoManagement;
/
```




```
CREATE OR REPLACE PACKAGE BODY ConvoManagement IS
   PROCEDURE GetStudentInfo(
      p_student_id IN updateinformation.S_ID%TYPE,
      p_student_name OUT updateinformation.S_NAME%TYPE,
      p_department OUT updateinformation.DEPARTMENT%TYPE
   ) IS
   BEGIN
      SELECT S_NAME, DEPARTMENT
      INTO p_student_name, p_department
      FROM updateinformation
      WHERE S_ID = p_student_id;

      EXCEPTION
         WHEN NO_DATA_FOUND THEN
            p_student_name := NULL;
            p_department := NULL;
   END GetStudentInfo;
END ConvoManagement;
/
```




```
DECLARE
   v_student_name updateinformation.S_NAME%TYPE;
   v_department updateinformation.DEPARTMENT%TYPE;
BEGIN
   ConvoManagement.GetStudentInfo(
      p_student_id => 17,
      p_student_name => v_student_name,
      p_department => v_department
   );

   DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
   DBMS_OUTPUT.PUT_LINE('Department: ' || v_department);
END;
/
```



Autocommit Display 10

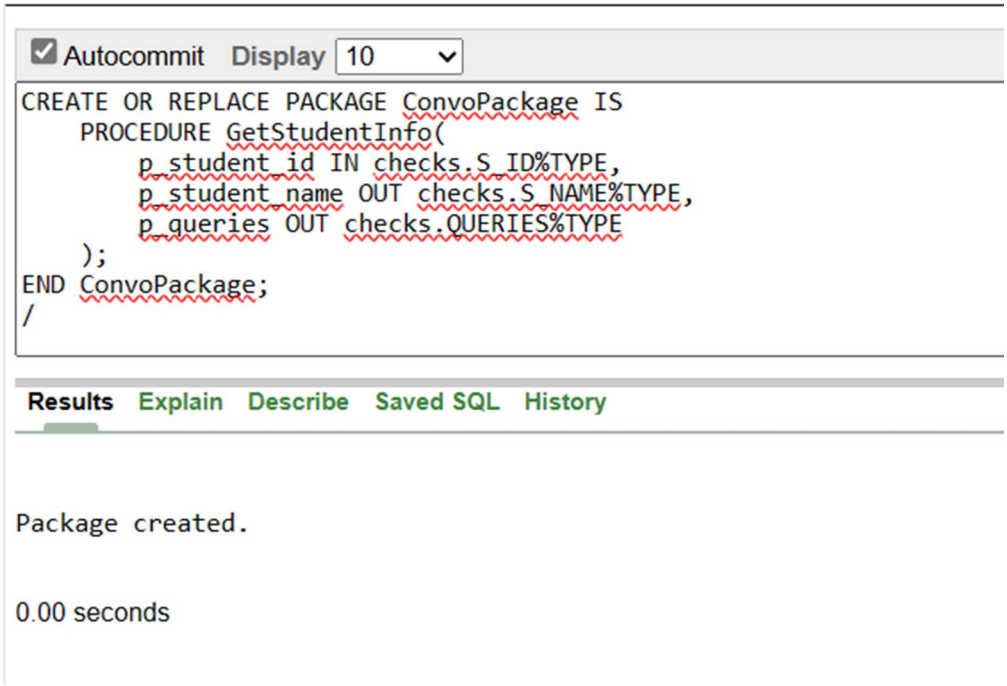
```
DECLARE
    v_student_name updateinformation.S_NAME%TYPE;
    v_department updateinformation.DEPARTMENT%TYPE;
BEGIN
    ConvoManagement.GetStudentInfo(
        p_student_id => 17,
        p_student_name => v_student_name,
        p_department => v_department
    );

    DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
    DBMS_OUTPUT.PUT_LINE('Department: ' || v_department);
END;
/
```

Results Explain Describe Saved SQL History

```
Student Name: Ron
Department: EEE

Statement processed.


0.00 seconds
```

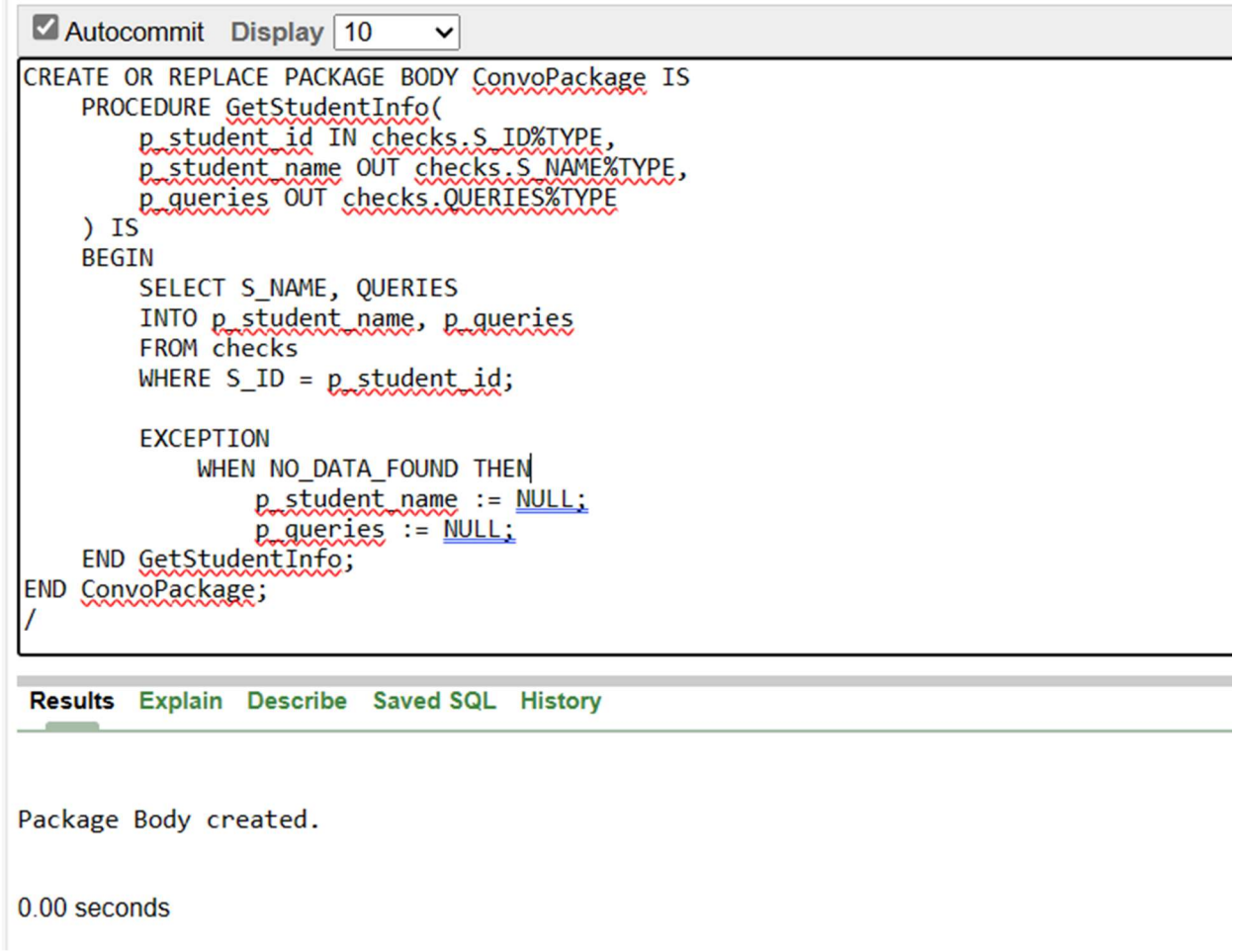
- Create a package on checks table to return student name and queries by SID ;

```
CREATE OR REPLACE PACKAGE ConvoPackage IS
  PROCEDURE GetStudentInfo(
    p_student_id IN checks.S_ID%TYPE,
    p_student_name OUT checks.S_NAME%TYPE,
    p_queries OUT checks.QUERIES%TYPE
  );
END ConvoPackage;
/
```




```
CREATE OR REPLACE PACKAGE BODY ConvoPackage IS
  PROCEDURE GetStudentInfo(
    p_student_id IN checks.S_ID%TYPE,
    p_student_name OUT checks.S_NAME%TYPE,
    p_queries OUT checks.QUERIES%TYPE
  ) IS
  BEGIN
    SELECT S_NAME, QUERIES
    INTO p_student_name, p_queries
    FROM checks
    WHERE S_ID = p_student_id;

    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        p_student_name := NULL;
        p_queries := NULL;
  END GetStudentInfo;
END ConvoPackage;
```

/




```
DECLARE
   v_student_name checks.S_NAME%TYPE;
   v_queries checks.QUERIES%TYPE;
BEGIN
   ConvoPackage.GetStudentInfo(
      p_student_id => 1,
      p_student_name => v_student_name,
      p_queries => v_queries
   );

   DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
```

```
        DBMS_OUTPUT.PUT_LINE('Queries: ' || v_queries);
    END;
/
```

Autocommit Display 10

```
DECLARE
    v_student_name checks.S_NAME%TYPE;
    v_queries checks.QUERIES%TYPE;
BEGIN
    ConvoPackage.GetStudentInfo(
        p_student_id => 1,
        p_student_name => v_student_name,
        p_queries => v_queries
    );

    DBMS_OUTPUT.PUT_LINE('Student Name: ' || v_student_name);
    DBMS_OUTPUT.PUT_LINE('Queries: ' || v_queries);
END;
/
```

Results Explain Describe Saved SQL History

```
Student Name: Ajoy
Queries: givemetheupdate

Statement processed.


0.00 seconds
```

Relational Algebra:

- Find the name of Student whose department is CSE.
  $\prod_{s\_name} (\sigma_{department= \text{“CSE”}} (updateinformation))$
- Find department and payment status where Student name is Mukit.
  $\prod_{department,paymentstatus} (\sigma_{s\_namer=\text{“Mukit”}}(checkaccounts))$
- Find student name and id whose Payment status is DONE.
  $\prod_{s\_name,s\_id} (\sigma_{paymentstatus=\text{“DONE”}}(checkaccounts))$
- Find the name of student whose Email is anika@gmail.com.

$\prod_{s\_name} (\sigma_{email=\text{“anika@gmail.com”}}(purchase))$

- Find the name and id of student whose queries is giveupdate.
  $\prod_{s\_name,s\_id} (\sigma_{queries=\text{“giveupdate”}}(check))$

Conclusion

The ABC is a very reputed and well-known university in Bangladesh. It has proposed a software system which will simplify all the complicated processes of traditional convocation management. It revolutionizes convocation management by leveraging technology to streamline processes, enhance coordination, and deliver a seamless experience for graduates and their families. By automating attendee registration which enables graduates to register for the convocation ceremony online, providing a convenient and hassle-free process, simplifying certificate management which automates the generation, verification, and distribution of graduation certificates. Graduates' academic records are seamlessly integrated into the system, allowing for accurate and timely certificate generation, facilitating event planning which enables the university administration to manage the venue selection, seating arrangements, audiovisual requirements, and other logistical aspects of the convocation ceremony, and providing insightful analytics which generates comprehensive reports on attendee registration, ticket sales, certificate distribution, and other convocation-related metrics. And finally, this system empowers the ABC university to host successful and memorable convocation ceremonies.