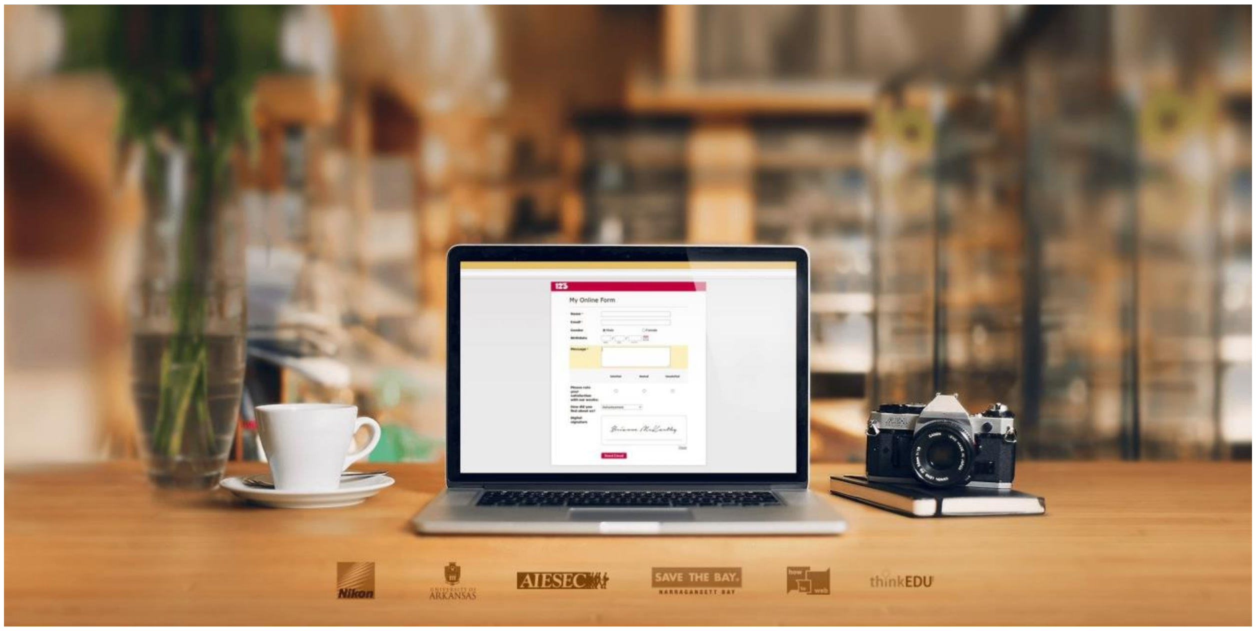


PROJECT REPORT

ON

University Student Information Management System

SUBMITED BY:

Mehreen Najm

Contents

Introduction 2
Acknowledgment 2
Abstract 3
Problem Statement 3
Purpose/Objective of Project 4
System Overview 4
  First Customer 4
  Terminology 4
System Description/Requirements 4
  Users & System Process 6
System Design 7
  Database Logical Design 7
  Database Physical Design 9
  Quires 11
    DDL Statements 11
    DML Statements 14
  PL/SQL packages for System Manipulation 16
References 19

Introduction

As we know in nowadays information technology is a very important and basic part to know it.

Researchers said a literate person must have four excellences:

- Writing
- Reading
- Listening
- Knowing computer

We can say if someone doesn't know about information technology he/she is an illiterate person.

Student information management system is a system that has the capabilities of storing and managing information about students and their related issues. Student Management System deals with all kind of student details, academic related reports, A Student Management System (SMS) is designed to help university for management of their students. Viewing student data, semester, fees payment and for examination, subject management, schedule, result and related issues are made simple and easy.

SIMS is plat-form it is useable in every OS such as: UNIX, LINUX, WINDOWS and WINDOWS SERVER.
SMIS can be web _based application, disktop_application.
This system contains three parts:

1st part is about the main purpose of this project why we create this system.

2nd part is about whole description of this system how it works and what we need in our system.

3rd part is about our system design their relationships and querying part.

I created this system by help of our kind teacher Spozhmay Ikhlas which she helped me in each step of creating and designing this system.

SIMS just used in universities such as: Khane Noor university and it is not useable and suitable for an organization, school or somewhere else.

Acknowledgment

First of all I am thankful from Almighty Allah for giving me this knowledge for papering and customizing this project. Second I am thankful from our guider Teacher Spozhmay "Ikhlas" for helping me to build the SIMS project.

This research was supported/partially supported by Rana University I am thankful to our colleagues who provided expertise that greatly assisted the research am also grateful from my family for guiding me in every step of life decisions.

Abstract

Student Management System (SMS) tracks all the details of a student from the day one to the end of his education which can be used for all reporting purpose, tracking enrolment, progress in the completed semesters years, final exam results and all these will be available for future references too.

Our project will have the databases of students offered by the university under all levels of graduation or main streams related to teacher or faculty details, students' details in all aspects. Different reports and Queries can be generated based of vast options related to students, batch, course, quota, semesters and category and even for the entire university.

This can make the system easier to navigate and to use maximizing the effectiveness of time and other resources. SMS allows the keeping of personnel data in a form that can be easily accessed and analyzed in a consistent way.

Problem Statement

As we see Afghanistan according absence of resort against information technology in past, it was faced with many problems that we can say the first problem was lack of resort to information Technology.

Lack of resort to IT was caused that it takes much time, lack of accessing to information on time.

If we solve this problem and create this system, universities in Afghanistan will be able to retrieve information on time less than a minute and their information will be always save and avoid from taking of much time.

As we see there are 3 main problems that caused to create SIMS system.

1. We don't have such kind of system for managing students' information.
2. In the past most of systems were paper-based.
3. Taking too much time.

We create SIMS manage students' information and it is a computerized system our data can't be loose, it will be save for a long-term and we can get needy information in a less time.

Purpose/Objective of Project

- SIMS project is about to handle, store and manage all the information of the students.
- To view, update, delete student's information by administrator.
- SIMS project is created to prevent from loosing of data.
- SIMS project is created to prevent from taking of too much time.

System Overview

SIMS is a system that enables you to store, manage, control, access students' information very easily.

SIMS has capabilities of:

- ➢ Adding, updating, deleting students' information and their related issues.
- ➢ Adding, updating, deleting teachers' information and their related issues.
- ➢ Adding, updating, deleting departments' information.
- ➢ Adding, updating, deleting users' information.

First Customer

SIMS is a system that is only used in universities.

We can't use this system in organizations, offices, supermarkets and etc.

SMIS is a system which is not belongs to a specific university; it is useable in many universities according to requirements.

Terminology

- ✓ SIMS (student information management system)
- ✓ DDL (data definition language)
- ✓ DML (data manipulation language)
- ✓ PL/SQL (procedural language/ standard query language)
- ✓ OS (operating system)

System Description/Requirements

Student information management system is a system that has the capabilities of storing and managing information about students and their related issues.

SIMS must have capabilities such as:

- Store, access , control , monitor , view , alter students' information
- Administrator with a unique username and password to manage the system.

System will be able to store information about students such as:

- ✓ Registration_number
- ✓ Father name
- ✓ Gender
- ✓ First_name
- ✓ Last_name
- ✓ Fees
- ✓ City
- ✓ Country
- ✓ photo
- ✓ Tazkira.no
- ✓ Department
- ✓ Faculty
- ✓ Semester
- ✓ Section
- ✓ Date of birth
- ✓ Email
- ✓ Address
- ✓ Phone number
- ✓ Next to kin name and number
- ✓ Class
- ✓ Subjects

System will be able to store information about students department and faculty such as:

- ✓ Faculty_name
- ✓ Department_name

System will be able to store information about teachers such as:

- ✓ First_name
- ✓ Last_name
- ✓ Salary
- ✓ Overtime
- ✓ City
- ✓ Country
- ✓ Photo
- ✓ Phone_number
- ✓ Address
- ✓ Email

System will be able to store information about users such as:

- ✓ First_name
- ✓ Last_name
- ✓ Phone_number
- ✓ Email_address
- ✓ Username

- ✓ password

Users & System Process

SIMS has 2 kinds of users:

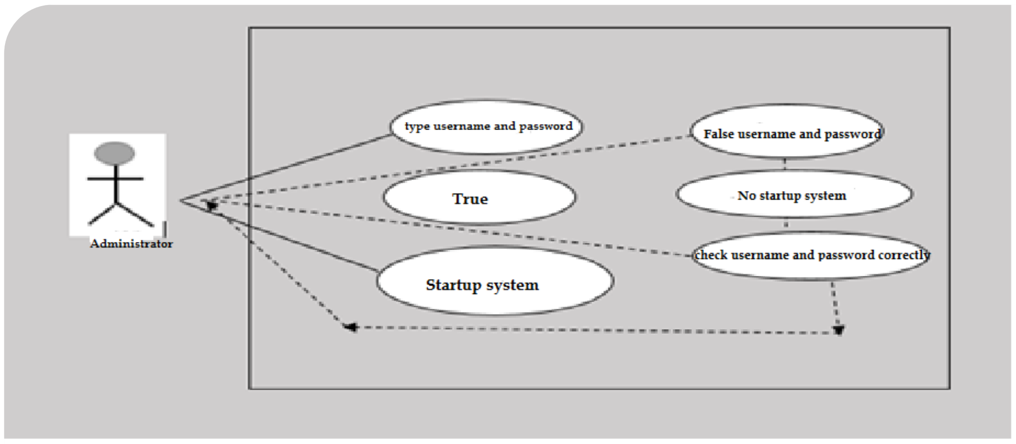
- Administrator (who can manage, monitor, organize the system)
- Other users (who can access , view information with privileges that administrator gave to them)

In order to provide a clearer picture of the system process provided by the student management module, we have done a use case analysis. Figure below is a use case diagram to present the system process in the student management module.

System Process:

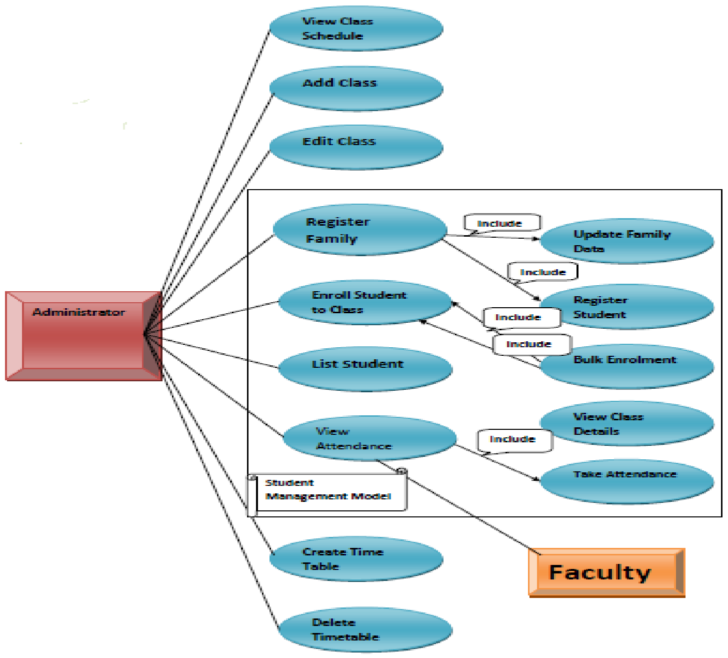
System process can be done by administrator who has ability to manage, control, monitor, organize and view whole system by typing the username and password correctly.

We can say administrator is a system user.


Administrator
type username and password
False username and password
True
No startup system
check username and password correctly
Startup system

this diagram show if user type his/her name correctly system will be startup and open up, otherwise if user type his/her username or password incorrect system will be not startup.

Now we provide other diagram that what can an administrator do when the system startup.



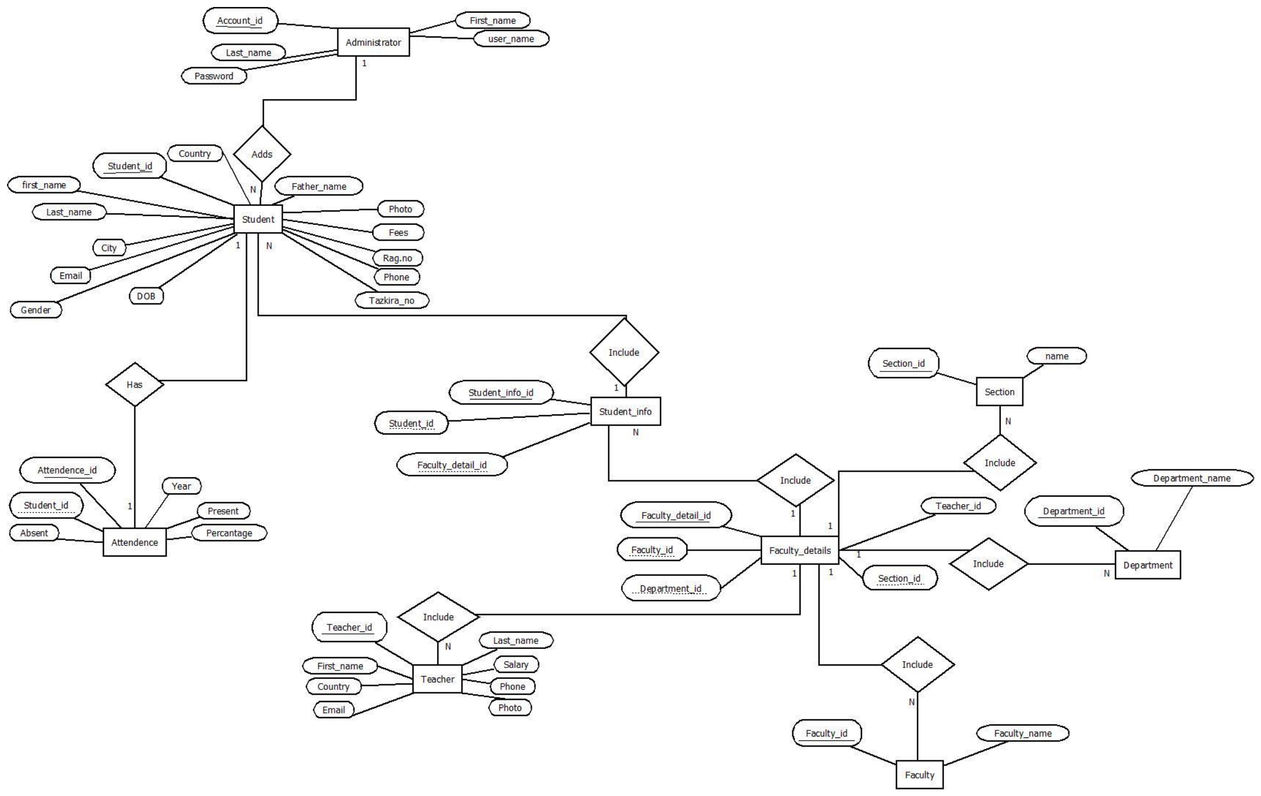
System Design

An entity–relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers are the physical implementation of the relationships.

Database Logical Design

Database logical design model would mean the schema would become as follows (the primary key and foreign key is underlined):



Now we are testing every relation against normal forms:

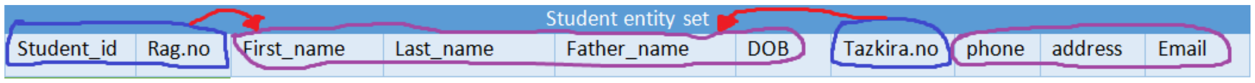
- ✓ A relation is considered to be in first normal form if all of its attributes have domains that are Indivisible or atomic.

Student entity set			
Attributes	**Type**	**Domain**	**Optional**
Student_id	Unique identifier		
Rag.no	Single value attribute	text	No
name	Single value attribute	text	No
last_name	Single value attribute	text	Yes
Father_name	Single value attribute	text	Yes
gender	Single value attribute	text	Yes
DOB	Single value attribute	text	No
class	Single value attribute	text	No
age	Single value attribute	text	Yes
Tazkira_no	Single value attribute	text	No
country	Single value attribute	text	Yes
city	Single value attribute	text	Yes
Email_address	Single value attribute	text	Yes
Phone	Multi value attribute	text	No
address	Single value attribute	text	Yes

So we should bring these multi_value attributes to Single_value attributes as bellow:

Phone	Single value attribute
address	Single value attribute

- ✓ A relation is in second formal form when it is in 1NF and there is no such non-key attribute that depends on part of the candidate key, but on the entire candidate key.



Student entity set									
Student_id	Rag.no	First_name	Last_name	Father_name	DOB	Tazkira.no	phone	address	Email

- ✓ A relation is in third normal form if it is in 2NF and there is no such non-key attribute that depends transitively on the candidate key. That is every attribute depends directly on the primary key. In our example, there is no such a situation; therefore, there is no reason to continue further with decomposition. All relations are in the 3rd normal form.

Database Physical Design

In Physical Design we are going to explain each table and specifying their columns and domains.

Admin Table: This table stores information about users and administrator of system.

Attributes	Type	Domain
Account_id	Unique identifier	text
First_name	Single value attribute	text
last_name	Single value attribute	text
username	Single value attribute	text
password	Single value attribute	text

Student Table: This table stores information about students and their related issues which is needed to be store.

Attributes	Type	Domain
Student_id	Unique identifier	**Text**
Rag.no	Single value attribute	text
name	Single value attribute	text
last_name	Single value attribute	text
Father_name	Single value attribute	text
gender	Single value attribute	text
DOB	Single value attribute	text
age	Single value attribute	text
Photo	Single value attribute	text
Tazkira_no	Single value attribute	text
country	Single value attribute	text
city	Single value attribute	text
Email_address	Single value attribute	text
Phone	Multi value attribute	text

address	Single value attribute	text
Fees	Single value attribute	text
semester	Single value attribute	text

Teacher Table: This table stores information about teachers and the related issues which is needed to be store.

Attributes	Type	Domain
Teacher_id	Unique identifier	text
First_name	Single value attribute	text
last_name	Single value attribute	text
gender	Single value attribute	text
phone	Single value attribute	text
Email_address	Single value attribute	text
Photo	Single value attribute	text
salary	Single value attribute	text

Attendance Table: This table stores information about attendance of students per year.

Attributes	Type	Domain
Attendance_id	Unique identifier	text
Student_id	Single value attribute	text
Absent	Single value attribute	number
present	Single value attribute	number
year	Single value attribute	number
percentage	Single value attribute	number

Section Table: This table stores information about section of each student.

Attributes	Type	Domain
Section_id	Unique identifier	text
Section_name	Single value attribute	text

Faculty Table: This table stores information about faculty which student belongs to it.

Attributes	Type	Domain
Faculty-id	Unique identifier	text
Faculty_name	Single value attribute	text

Department Table: This table stores information about department of each faculty and student which belongs to it.

Attributes	Type	Domain
Department_id	Unique identifier	text
dep_name	Single value attribute	text

Student_info Table: This table stores required information about student, his/her faculty , his/her department name, his/her teacher , his/her section.

Attributes	Type	Domain
Student_info_id	Unique identifier	text
Student_id	Single value attribute	text
Faculty_detail_id	Single value attribute	text

Faculty_details: Table: This table stores information about section name, teacher information, faculty names, department names and so on.

Attributes	Type	Domain
Faculty_detail_id	Unique identifier	text
Teacher_id	Single value attribute	text
Section_id	Single value attribute	text
Department_id	Single value attribute	text
Faculty_id	Single value attribute	text

Quires:

To implement our Logical Design to Physical Design we should use from SQL.

DDL Statements:

First I want to create Sequence for our unique identifiers to increment their numbers automatically.

Syntax of creating a Sequence:

```
CREATE SEQUENCE n_number

NOCACHE;
```

Now we are going to create our tables with their constraints and columns:

Creating student table:

```
CREATE TABLE student (

Student_id INTEGER  CONSTRAINT student_pk PRIMARY KEY,

First_name VARCHAR(12)  NOT NULL,

Last_name VARCHAR(12) NOT NULL,

Father_name VARCHAR(15) NOT NULL,

Photo VARCHAR(800),

Tazkira_no VARCHAR(20) UNIQUE,
gender VARCHAR(9),

dob VARCHAR(30),

age VARCHAR(20),

fees VARCHAR(30),

city VARCHAR(20) ,

country VARCHAR(20),

phone_number VARCHAR(25)  NOT NULL,

email_address VARCHAR(30),

address VARCHAR(30),

semester VARCHAR(3));
```

Creating teacher table:

```
CREATE TABLE teacher (

Teacher_id INTEGER CONSTRAINT teacher_pk PRIMARY KEY,

First_name VARCHAR(12)  NOT NULL,

Last_name VARCHAR(12) NOT NULL,

Gender VARCHAR(9),

Phone_number VARCHAR(20),

Salary VARCHAR(30),

Email_address VARCHAR(30),

Photo VARCHAR(700));
```

Creating attendance table:

```
CREATE TABLE attendance (

Attendance_id INTEGER CONSTRAINT attendance_pk PRIMARY KEY,

Student_id NUMBER(7) CONSTRAINT student_fk REFERENCES student(student_id),

Present VARCHAR(20),

Absent VARCHAR(20),

Year VARCHAR(15),

Percentage VARCHAR(20));
```

Creating faculty table:

```
CREATE TABLE faculty (
Faculty_id INTEGER CONSTRAINT faculty_pk PRIMARY KEY,
Faculty_name VARCHAR(20));
```

Creating department table:

```
CREATE TABLE department (
Department_id INTEGER CONSTRAINT dep_pk PRIMARY KEY,
Department_name VARCHAR(20));
```

Creating Section table:

```
CREATE TABLE section (
Section_id INTEGER CONSTRAINT section_pk PRIMARY KEY,
Section_name VARCHAR(20));
```

Creating faculty_details table:

```
CREATE TABLE faculty_details (
Faculty_detail_id INTEGER CONSTRAINT f_pk PRIMARY KEY,
Teacher_id  NUMBER(7) CONSTRAINT teacher_id_fk REFERENCES teacher(teacher_id),
department_id NUMBER(7) CONSTRAINT dep_id_fk REFERENCES department(department_id),
faculty_id NUMBER(7) CONSTRAINT faculty_id_fk REFERENCES faculty(faculty_id),
section_id NUMBER(7) CONSTRAINT section_id_fk REFERENCES section(section_id));
```

Creating student_info_table:

```
CREATE TABLE student_info (
Student_info_id INTEGER CONSTRAINT info_pk PRIMARY KEY,
student_id NUMBER(7) CONSTRAINT student_id_fk REFERENCES student(student_id),
faculty_detail_id NUMBER(7) CONSTRAINT detail_fk REFERENCES faculty_details(faculty_detail_id));
```

Altering a constraint:

ALTER TABLE student

ADD CONSTRAINT chek_gender CHECK (gender IN ('MALE' , 'FEMALE'));

- ✓ Now we are creating views using sub queries:

  CREATE OR REPLACE VIEW view_name AS
  (SUBQUERY WHERE CONDITION);

  CREATE OR REPLACE VIEW student_backup AS
  (SELECT student_id,first_name,last_name FROM student WHERE phone_number LIKE '079%';);

DML Statements:

- To insert data in tables we use from this syntax:
  INSERT INTO table_name (CULOMN1, CULOMN2,...) VALUES (VALUE1, VALUE2,...);

  - ❖ INSERT INTO student VALUES (n_number.nextval, 'mehreen' , 'najm', 'sarajudine' , 'C:\Users\Public\Pictures\Sample Pictures' , 99897 , 'FEMALE' , '4-15-1997' , 19 , 16000 , 'kabul' , 'afghanistan' , '079897382' , 'mehreenhushmand@gmail.com' , '2th Macroyan ' , 5);

  - ❖ INSERT INTO student VALUES (n_number.nextval, 'Sumaya' , 'fahimi', 'abdul wahid' , 'C:\Users\Public\Pictures\sumaia.png' , 97454 , 'FEMALE' , '4-15-1994' , 23 , 18000 , 'kabul' , 'afghanistan' , '07873983' , 'somaiafahimi23@gmail.com' , 'Baraki_square ' , 3);

  - ❖ INSERT INTO student VALUES (n_number.nextval, 'Tamim' , 'Ahmadi', 'Hayatullah' , 'C:\Users\Public\Pictures\tamim.png' , 97834 , 'MALE' , '4-15-1996' , 19 , 15000 , 'herat' , 'afghanistan' , '07990893' , 'tamimahmadi123@gmail.com' , 'Karte-naw ' , 7);

- ➢ INSERT INTO teacher VALUES (1, ‘Spozhmay’ , ‘Ikhlas’, ‘FEMALE’ , ‘078978383’ , 10000 , ‘Spozhmayikhlas143@gmail.com’ , ‘Libraries\Pictures’);

- ✓ INSERT INTO attendance VALUES (1, 1 , 355, 5 , ‘97%’);
- ✓ INSERT INTO faculty VALUES (1, ‘Computer science’);
- ✓ INSERT INTO department VALUES (1,’Database’);
- ✓ INSERT INTO section VALUES (1, ‘A_DBA’);
- ✓ INSERT INTO faculty_details VALUES (1, 1, 1, 2, 1);
- ✓ INSERT INTO Student_info VALUES (1, 1);

- To view data in a table use SELECT statement:
  SELECT (CULOMN1, CULOMN2, …) FROM TABLE_NAME;
  OR
  SELECT * FROM TABLE_NAME WHERE CONDATION;

- To update data in a table use this syntax:

  UPDATE TABLE_NAME SET CULOMN_NAME = CONDITION;

  UPDATE student SET fees = 12000 WHERE student_id <5;

- To delete data from a table use this syntax:
  DELETE FROM table_name;
  DELETE FROM teacher;
  OR
  DELETE FROM table_name WHERE CONDATION;
  DELETE FROM students WHERE first_name LIKE ‘%e’;

- To get data from 2 or more than two tables use joins:
  SELECT table_name1.culomn_name, table_name2.column_name, …
  FROM table_name1 [INNER JOIN, OUTER JOIN, JOIN] table_name2
  ON (table_name2.column_name = table_name2.column_name);

PL/SQL packages for System Manipulation

. Create a PL/SQL package as following:

a. Package would have a function named average_high_fees_student which will return the students first_name whom pays the average fees.

b. Package would have a function named high_salary which will return the teachers first_name and id whom receive the high salary.

c. Package would have a function named low_salary which will return the teachers first_name and id whom receive the low salary.

d. The package would also have a procedure using which we can insert new students to the student table.

e. The package would also have a procedure using which we can update the student fees.

ANSWER:

1st: we must create package specification like below:

```
CREATE OR REPLACE PACKAGE student_package

AS

    TYPE s_cursor IS REF CURSOR;

FUNCTION average_high_fees_student RETURN s_cursor;

FUNCTION high_salary RETURN s_cursor;

FUNCTION low_salary RETURN s_cursor;


PROCEDURE insert_new_student(

    std_id IN student.student_id%TYPE,


    std_first_name IN student.first_name%TYPE,

    std_last_name IN student.last_name%TYPE,

    std_father_name IN student.father_name%TYPE

  std_city IN student.city%TYPE,

  std_country IN student.country%TYPE,

  std_photo IN student.photo%TYPE,

 std_phone IN student.phone_number%TYPE,
  std_tazkira_no IN student.tazkira_no%TYPE,);


PROCEDURE update_student_fees(
    s_id IN student.student_id%TYPE,
    std_salary IN INTEGER );
END student_package;
/
```

#2nd:

We must create package body like below:

```
CREATE OR REPLACE PACKAGE BODY student_package
AS
FUNCTION average_high_fees_student RETURN s_cursor
IS
v_average s_cursor;
BEGIN
OPEN v_average FOR
SELECT first_name FROM student WHERE
fees>(SELECT AVG(fees FROM student);
RETURN v_average;
END average_high_fees_student;
FUNCTION high_salary RETURN s_cursor
IS
v_high s_cursor;
BEGIN
OPEN v_high FOR
SELECT teacher_id,first_name FROM teacher WHERE
salary = (SELECT MAX(salary) FROM teacher);
RETURN v_high;
END high_salary;
```

```
FUNCTION low_salary RETURN s_cursor

IS

v_low s_cursor;

BEGIN

OPEN v_low FOR

SELECT teacher_id,first_name FROM teacher WHERE

salary>(SELECT MIN(salary) FROM teacher);

RETURN v_low;

END low_salary;

PROCEDURE insert_new_student(

      std_id IN student.student_id%TYPE,

      std_first_name IN student.first_name%TYPE,

      std_last_name IN student.last_name%TYPE,

      std_father_name IN student.father_name%TYPE

    std_city IN student.city%TYPE,

   std_country IN student.country%TYPE,

   std_photo IN student.photo%TYPE,

  std_phone IN student.phone_number%TYPE,

  std_tazkira_no IN student.tazkira_no%TYPE,)

AS

BEGIN

INSERT INTO student(student_id,
,first_name,last_name,father_name,city,country,photo,phone_number,tazkira_no)

VALUES(std_id, std_first_name,std_last_name,std_father_name,std_city, std_country, std_photo,
std_phone, std_tazkira_no);

COMMIT;

END insert_new_student;

PROCEDURE update_student_fees(

      s_id IN employees.employee_id%TYPE,
        s_salary IN INTEGER )

    AS

    BEGIN

    UPDATE student set fees = fees – s_salary

    WHERE student_id = s_id;

    COMMIT;

    END update_student_fees;

    END student_package;

    /
```

TO call package element use this Syntax:

```
Exec package_name.element_name;
```

OR

```
SELECT package_name.element_name FROM DUAL;
```

References

1. Natalie Gagliordi, "US universities at greater risk for security breaches than retail and healthcare: Bit Sight," ZDNet, August 21, 2014. Accessed at http://www.zdnet.com/us-universities-at-greater-risk-for-security-breaches-than-retail-and-healthcare-bitsight-7000032843/
2. Gary Langsdale, "What It Takes to Keep Student Information Safe in the Digital Age," The Evolution, October 2, 2014. Accessed at http://www.evolllution.com/opinions/audio-takes-student-information-safe-digital-age/
3. *"We Built, We Bought, We Shared: The Costs of Administrative Service Systems vs. the Academic Mission (EDUCAUSE Review) | EDUCAUSE.edu". www.educause.edu. Retrieved 2015-06-02.*
4. U.S. Department of Education Office of Planning, Evaluation and Policy Development (2009). *Implementing data-informed decision making in schools: Teacher access, supports and use.* United States Department of Education (ERIC Document Reproduction Service No. ED504191)
5. Rankin, J. (2013, March 28). How data Systems & reports can either fight or propagate the data analysis error epidemic, and how educator leaders can help. *Presentation conducted from Technology Information Center for Administrative Leadership (TICAL) School Leadership Summit.*