**Computer Science Department**

**University of Computer & Emerging Sciences (FAST-NU)**

HOME WORK ASSIGNMENT COVER SHEET

COURSE TITLE DATABASE SYSTEMS COURSE CODE CS204

INSTRUCTOR DR. EJAZ AHMED TYPE √ (Please tick)

Indiv. Group

ASSIGNMENT NO

ASSIGNMENT Lab Project – Hamarey Bachchey System

HAND OUT DATE 23-APR-2020 DUE DATE 15-MAY2020(Acceptable till 5-JUNE-2020)

ESTIMATE TIME 50 Hours

|  |  |
| --- | --- |
| ASSESSMENT CRITERIA (or attached) | % Mark |
| Instructions to Candidate:  Grade distribution is Attached  Subission: Print & online  Tools: **Oracle Database & PHP**  Group of max 4 students is allowed, individual work is discouraged with 10% deduction |  |

|  |
| --- |
| TO BE COMPLETED BY STUDENT (TEAM LEAD) GROUP MEMBERS ID  ID 18I-0737 Sec# D  ID 18I-0621 Sec# D  ID 18I-0731 Sec# D  ID 18I-474 Sec# D  NAME Haroon Ali  ID NO 18I-0737 SECTION: CS-D  Time Taken  DECLARATION: I/We declare that this Coursework is my/our group’s own work  SIGNATURES (All members) |

**Introduction:**

**The task provided to us was to design and develop a database for an NGO ‘Hamary Bachchey’. First of all, we analyzed the system requirements for this given project. Then we made few assumptions regarding the design. According to those assumptions we represented the requirement into logical design using ER model.**

**We split the workflow in the following steps:**

1. Read the document thoroughly and make some assumptions
2. Based on those assumptions, make ER diagram
3. From ER diagram, make a relational schema
4. From relational schema we made tables using SQL DML and DDL
5. Then we added the datasets into those tables.
6. Make required queries
7. Use PHP and Oracle to create UI of Forms and Reports

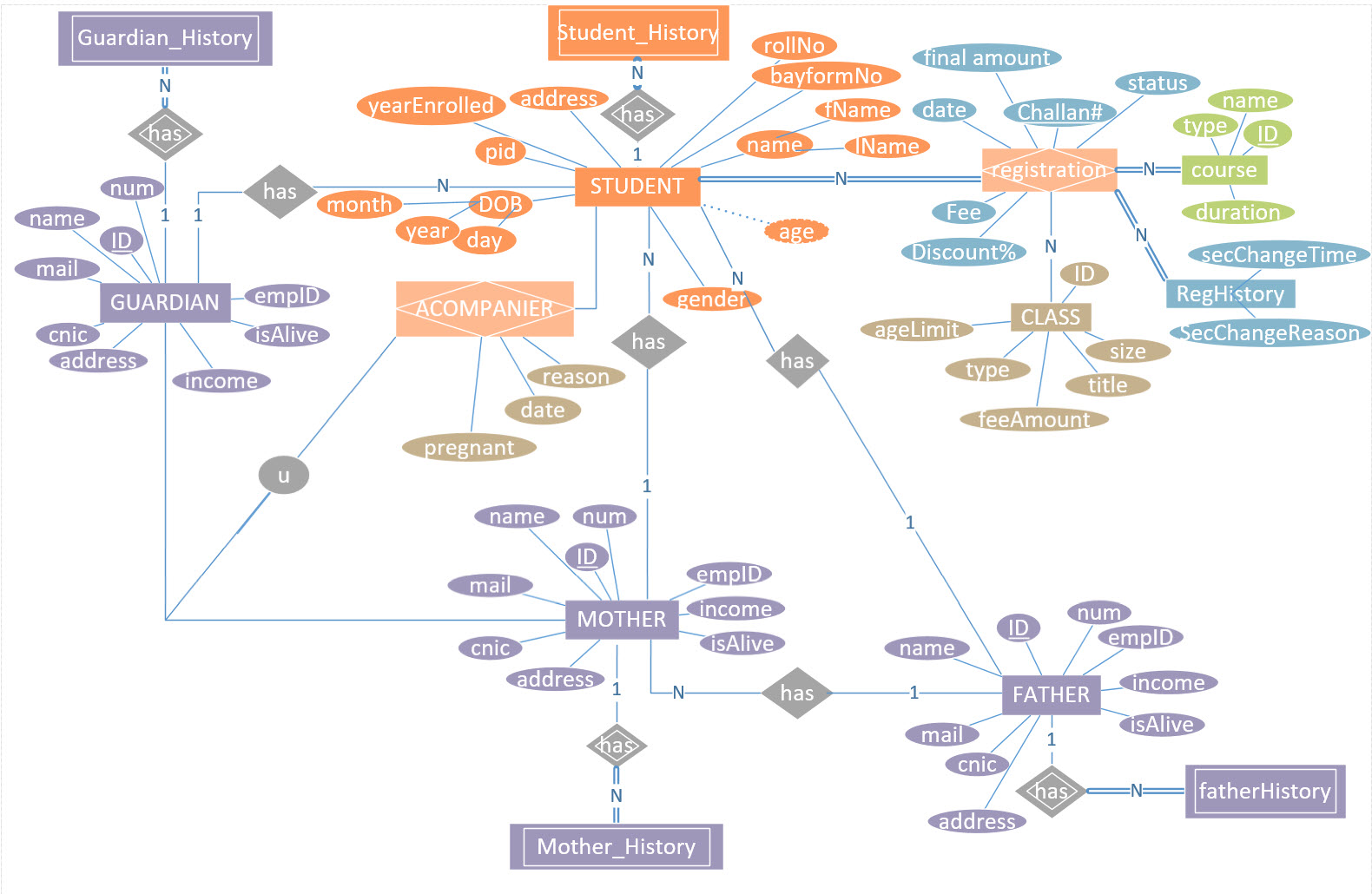
**Assumptions:**

After reading the document few times, we grasp the following main points. Moreover, we made some assumptions from ourselves, which we think were essential for our ER diagram and relational schema.

1. We made assumptions that we will be given father, mother, and guardian information no matter what. So, we made separate entities of father, mother and guardian.
2. One of either could accompany the child either the mother or the female guardian.
3. Fee is only paid once the course is registered.
4. Whenever a tuple of student, guardian, mother or father is changed, it is stored in their History table which is made dependent.
5. Fee is calculated with the discount in mind,
6. if child has more than 3 siblings, we will give some percentage of discount
7. if its employee child, we will give full discount.
8. Sections are made dynamic; in a manner that new sections are made automatically when a certain limit is crossed in one class.

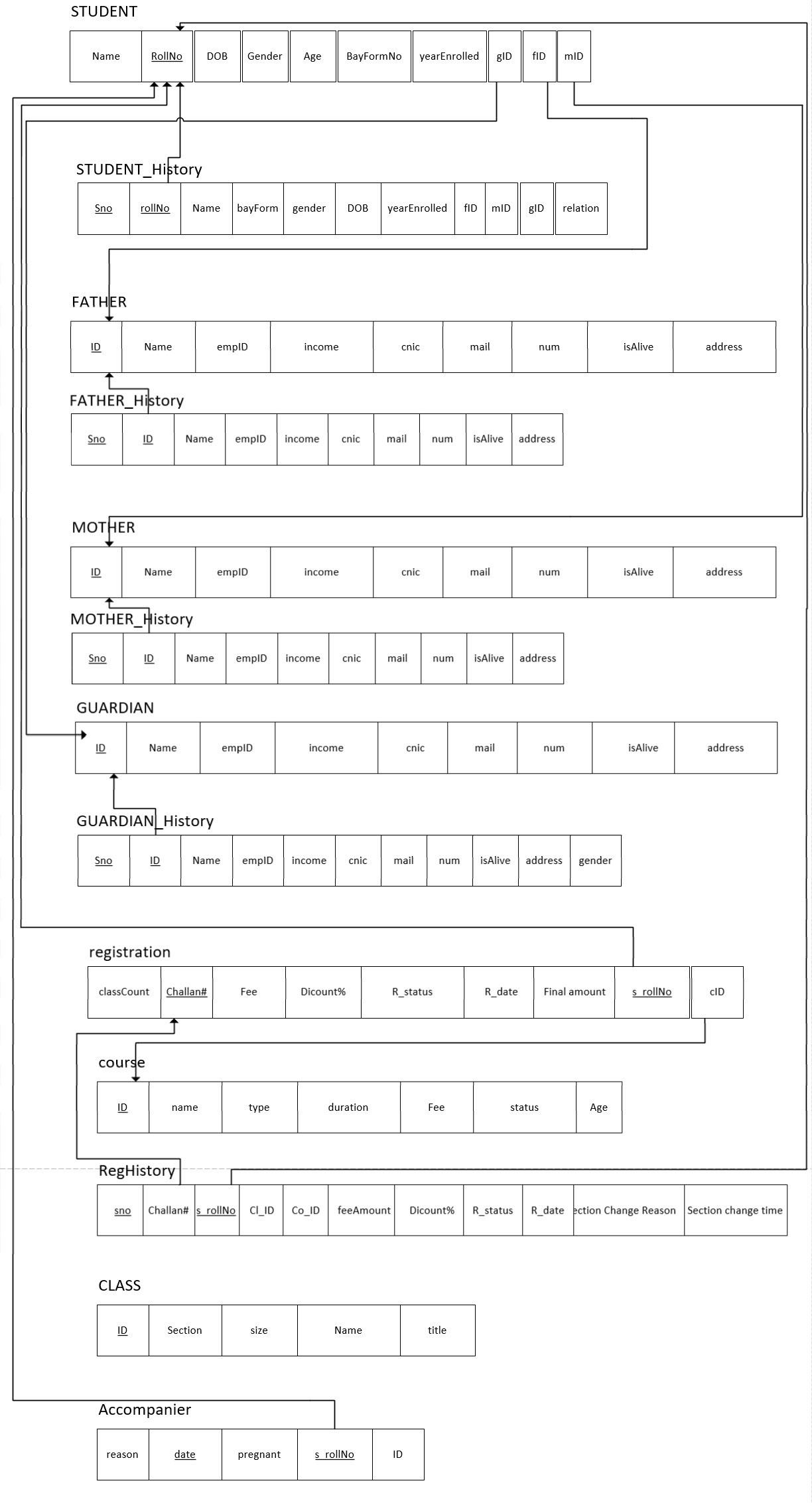
**Entity Relationship Diagram:**

After brainstorming all the main points and assumptions, we made a simple Entity Relationship Diagram (ERD). Which we revised later and made few changes here and there. After few iterations, the final version looks like this.



**Schema Diagram:**

When we finalized our ER diagram, we started working on blueprint of a database that outlines the way its structure organizes data into tables.



**Table Descriptions:**

STUDENT:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| Name | Name of the student | NOT NULL | Varchar2(30) |
| RollNo | Primary key of table | INDEX | Varchar2(9) |
| DOB | Derived attribute date of birth | Date, Not Null | Date |
| Gender | Gender of the student | M or F, NOT NULL | Varchar2(1) |
| Age | Age of the student | NOT NULL | Date |
| BayformNo | Bayform No provided by NADRA | UNIQUE | Varchar2(13) |
| yearEnrolled | Year since student is studying | NOT NULL | Date |
| gID | FK from table GUARDIAN | NOT NULL | Varchar2(9) |
| fID | FK from table FATHER | NOT NULL | Varchar2(9) |
| mID | FK from table MOTHER | NOT NULL | Varchar2(9) |

STUDENT\_History:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| sno | Serial nomber | INDEX | Number |
| Name | Name of the student | NOT NULL | Varchar2(30) |
| RollNo | FK from STUDENT table | INDEX | Number(12) |
| DOB | Derived attribute date of birth | Date, NOT NULL | Date |
| Gender | Gender of the student | M or F, NOT NULL | Varchar2(1) |
| Age | Age of the student | NOT NULL | Date |
| BayformNo | Bayform No provided by NADRA | NOT NULL | Number(12) |
| yearEnrolled | Year since student is studying | Date, NOT NULL | Date |
| gID | Foreign key from table Guardian | NOT NULL | Varchar2(9) |
| fID | Foreign key from table Father | NOT NULL | Varchar2(9) |
| mID | Foreign key from table Mother | NOT NULL | Varchar2(9) |
| relation | Relation with guradian | NOT NULL | varchar2(10) |

FATHER:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| ID | Primary key for this table | PK | Varchar(9) |
| Name | Name of the student | NOT NULL | Varchar(30) |
| empID | Unique employee id | NOT NULL | Varchar(30) |
| DOB | Derived attribute date of birth | Date, NOT NULL | date |
| income | Yearly income | NOT NULL | number(10) |
| cnic | Unique identity number | UNIQUE, NOT NULL | Varchar(9) |
| mail | Email address | NOT NULL | Varchar(9) |
| num | Phone number | NOT NULL | Number(11) |
| isAlive | Whether father is alive or dead | Yes or No, NOT NULL | Varchar2(1) |
| address | Physical, permanent address | NOT NULL | Varchar(100) |

FATHER\_History:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| sno | Serial number | NOT NULL | Number |
| ID | Foreign key from table FATHER | UNIQUE | Varchar2(9) |
| Name | Name of the student | NOT NULL | Varchar2(30) |
| empID | Unique employee id | UNIQUE, NOT NULL | Varchar2(30) |
| income | Yearly income | NOT NULL | Number |
| cnic | Unique identity number | UNIQUE, NOT NULL | Varchar2(13) |
| mail | Email address | UNQIUE, NOT NULL | Varchar2(50) |
| num | Phone number | NOT NULL | Number |
| isAlive | Whether father is alive or dead | Yes or No, NOT NULL | Varchar2(100) |
| address | Physical, permanent address | NOT NULL | Number |

MOTHER:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| ID | Primary key for this table | UNIQUE, PK | Varchar(9) |
| Name | Name of the student | NOT NULL | Varchar(30) |
| empID | Unique employee id | NOT NULL | Varchar(30) |
| DOB | Derived attribute date of birth | Date, NOT NULL | date |
| income | Yearly income | NOT NULL | number(10) |
| cnic | Unique identity number | UNQIUE, NOT NULL | Varchar(9) |
| mail | Email address | UNQIUE, NOT NULL | Varchar(9) |
| num | Phone number | NOT NULL | Number(11) |
| isAlive | Whether father is alive or dead | Yes or No, NOT NULL | Varchar2(1) |
| address | Physical, permanent address | NOT NULL | Varchar(100) |

MOTHER \_History:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| sno | Serial number |  | Number |
| ID | Foreign key from table MOTHER | UNQIUE, NOT NULL, PK | Varchar2(9) |
| Name | Name of the student | NOT NULL | Varchar2(30) |
| empID | Unique employee id | UNQIUE, NOT NULL | Varchar2(30) |
| income | Yearly income | NOT NULL | Number |
| cnic | Unique identity number | UNQIUE, NOT NULL | Varchar2(13) |
| mail | Email address | UNQIUE, NOT NULL | Varchar2(50) |
| num | Phone number | NOT NULL | Number |
| isAlive | Whether father is alive or dead | Yes or No, NOT NULL | Varchar2(100) |
| address | Physical, permanent address | NOT NULL | Number |

GUARDIAN:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| ID | Primary key for this table | PK, UNQIUE | Varchar(9) |
| Name | Name of the student | NOT NULL | Varchar(30) |
| empID | Unique employee id | UNQIUE, NOT NULL | Varchar(30) |
| DOB | Derived attribute date of birth | Date, NOT NULL | date |
| income | Yearly income | NOT NULL | number(10) |
| cnic | Unique identity number | UNQIUE, NOT NULL | Varchar(13) |
| mail | Email address | UNQIUE, NOT NULL | Varchar(9) |
| num | Phone number | NOT NULL | Number(11) |
|  |  |  |  |
| address | Physical, permanent address | NOT NULL | Varchar(100) |

GUARDIAN \_History:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| sno | Serial number | UNQIUE, | Number |
| ID | Foreign key from table GUARDIAN | UNQIUE | Varchar2(9) |
| Name | Name of the student | NOT NULL | Varchar2(30) |
| empID | Unique employee id | NOT NULL | Varchar2(30) |
| income | Yearly income | NOT NULL | Number |
| cnic | Unique identity number | UNQIUE, NOT NULL | Varchar2(13) |
| mail | Email address | UNQIUE, NOT NULL | Varchar2(50) |
| num | Phone number | NOT NULL | Number |
| address | Physical, permanent address | NOT NULL | Varchar2(100) |

Registration:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| classCount | Total strength of the class | NOT NULL | Number |
| Challan# | Primary key for this table | UNQIUE | Varchar2(9) |
| Fee | One time payable fee | NOT NULL | Number(5) |
| Discount% | Discount a student got due to some special status | NOT NULL | Number(3) |
| R\_Status | Show whether the student is registered or not | NOT NULL | Varchar2(10) |
| R\_date | Date at which the student is registered | Date, NOT NULL | date |
| finalAmount | Total final amount paybable after calculation | NOT NULL | Number(5) |
| S\_rollNo | foreignKey from student table | NOT NULL | Varchar2(9) |
| cID | Foreign key from course table | NOT NULL | Varchar2(9) |

COURSE:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| ID | Primary key for this table | PK | Varchar2(9) |
| Name | Name of the course | NOT NULL | Varchar2(30) |
| type | Category the course belongs to | NOT NULL | Carchar2(10) |
| duration | Duration of the course in months | NOT NULL | Varchar2(10) |
| Fee | Fee for this course | NOT NULL | Number(5) |
| status | Whether the course is currently available or not | NOT NULL | Varchar2(10) |
| Age | Age group this course is thought to | NOT NULL | Number(3) |

Reg \_History:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| sno | Serial number | PK | Number |
| Challan# | FK from Registration table | FK | Varchar2(9) |
| RollNo | Foreign key from student table | UNQIUE, NOT NULL | Varchar2(9) |
| Cl\_ID | Foreign key from class table | NOT NULL | Number |
| Co\_ID | Foreign key from course table | NOT NULL | Varchar2(9) |
| feeAmount | Total fee student have to submit | NOT NULL | Number(5) |
| Discount% | Discount a student got due to some special status | NOT NULL | Number(3) |
| R\_status | Registration status | NOT NULL | Varchar2(200) |
| R\_date | Registration date | Date, NOT NULL | Date |
| Section change reason | Reason why the student changed the section |  | Varchar2(200) |
| Section change time | When the student changed the section | Date, NOT NULL | date |

CLASS:

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| ID | Primary key for this table | PK | Number(10) |
| Section | Class section | NOT NULL | Varchar(1) |
| size | Total strength of the class | NOT NULL | Number(4) |
| Title | Title of the class | NOT NULL | Varchar(20) |

Accompanier:

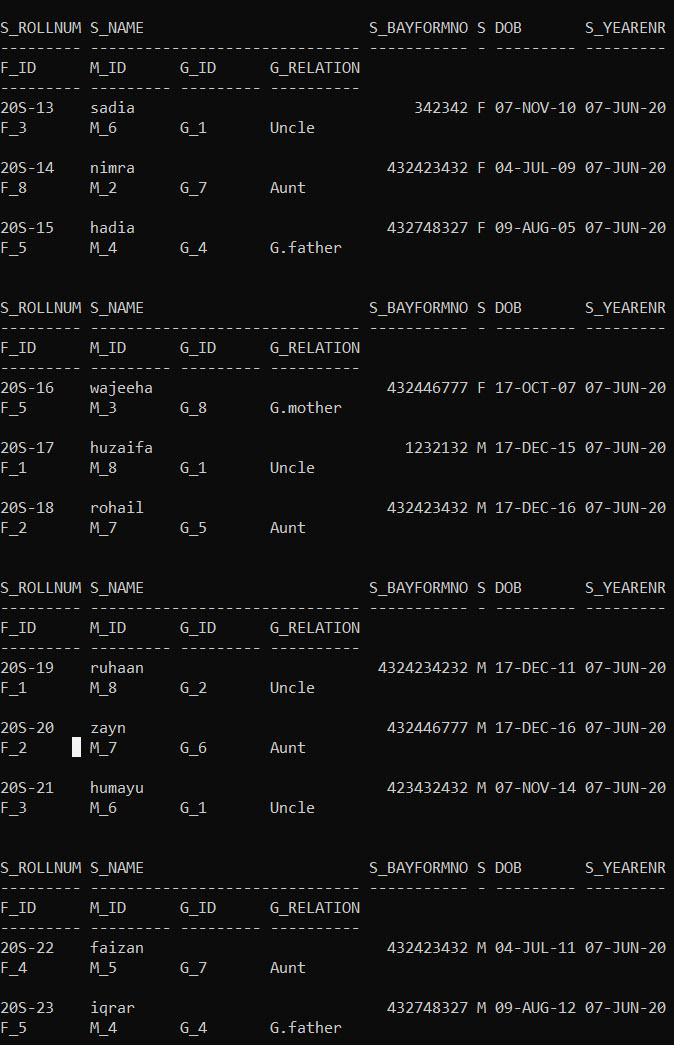
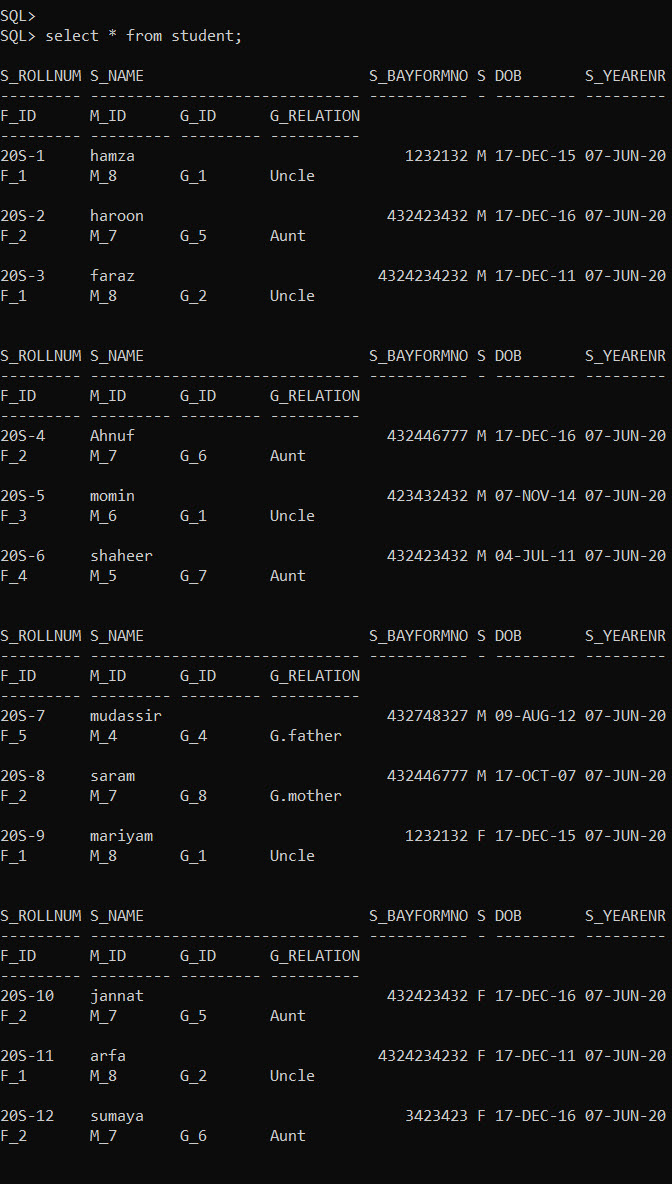
|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Constraints | Type |
| ID | Primary key for this table | PK | Varchar2(9) |
| S\_rollNo | Foreign key from student table | UNQIUE, NOT NULL | Varchar2(9) |
| date | When parent got accompanied with the student | Date, NOT NULL | Date |
| reason | Why parent accompanied student | NOT NULL | Varchar2(50) |
| pregnant | Whether she’s pregnant or not | NOT NULL | Varchar2(10) |

**Queries:**

After creation of database we filled it with 32 students, here are output of some of the queries we ran on SQL Plus.

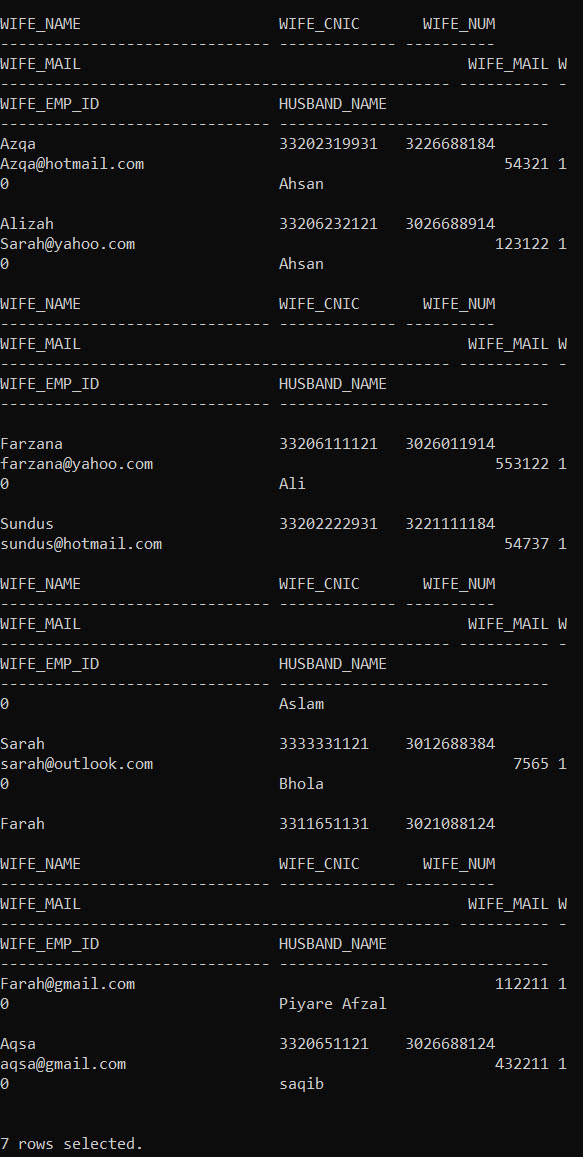
--query1

select \* from student;



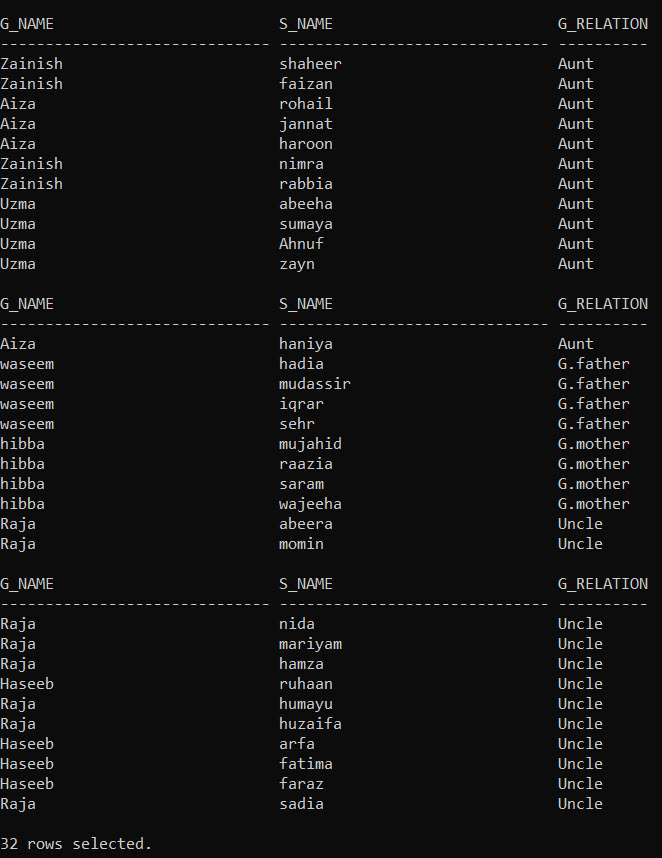
--query 2

select distinct m.m\_name wife\_name,m.m\_cnic wife\_cnic,m.m\_num wife\_num,m.m\_mail wife\_mail,m.m\_income wife\_mail,m.m\_isalive wife\_isalive,m.m\_emp\_id wife\_emp\_id,f.f\_name husband\_name from student s,father f,mother m where s.f\_id=f.f\_id and s.m\_id = m.m\_id order by husband\_name;



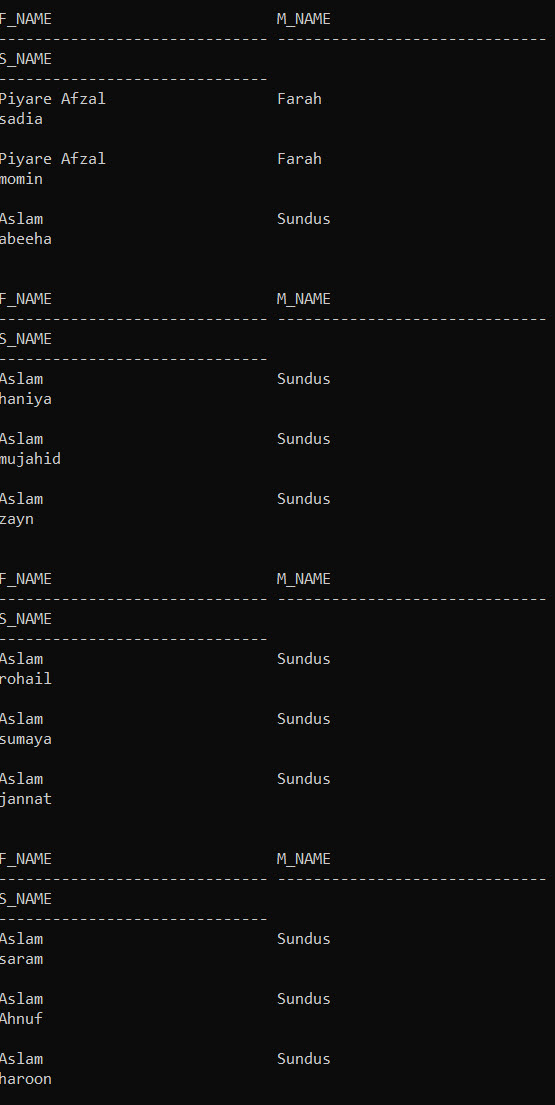
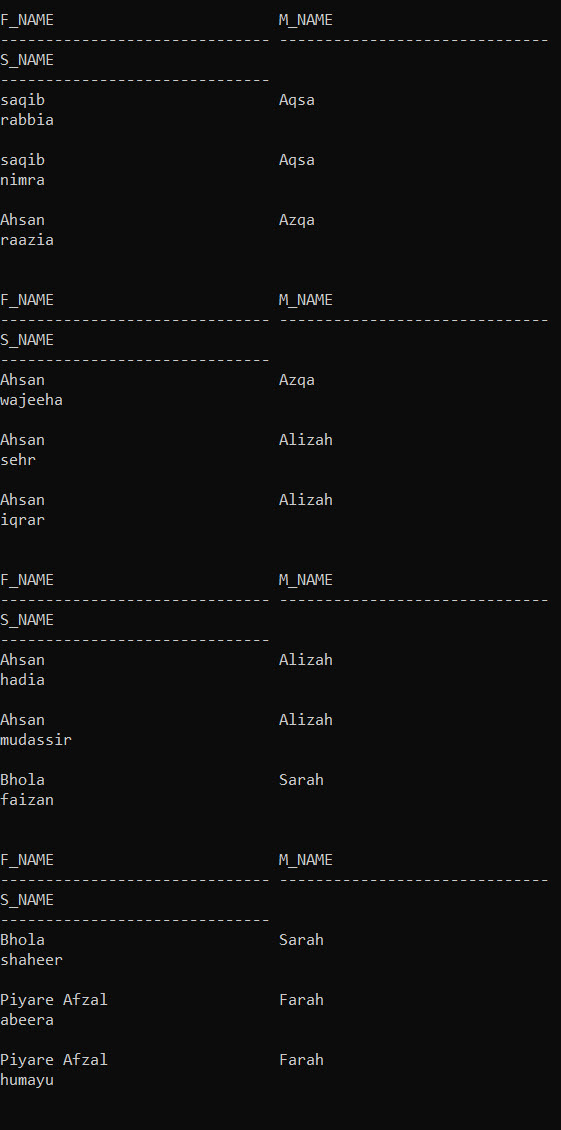
--query 3

select g.g\_name,s\_name,g\_relation from student s, guardian g where s.g\_id = g.g\_id order by g\_relation;



--query 4

select f.f\_name,m.m\_name,a.s\_name from student a, father f, mother m where a.f\_id = f.f\_id and a.m\_id = m.m\_id;



--query 5

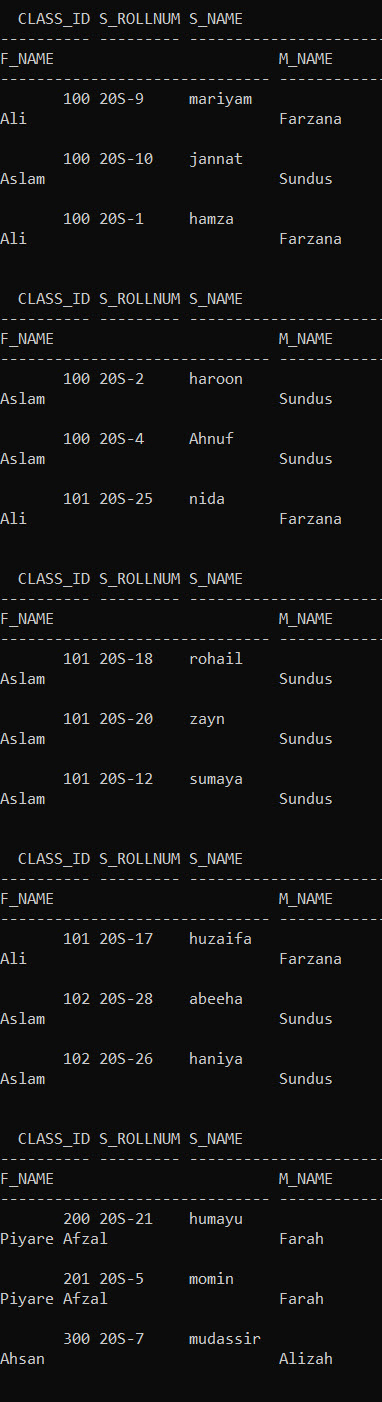
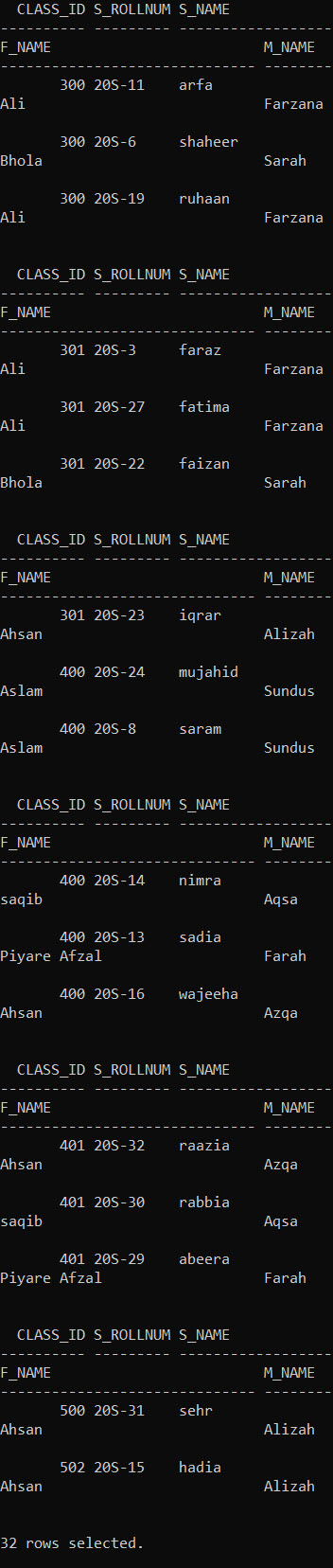
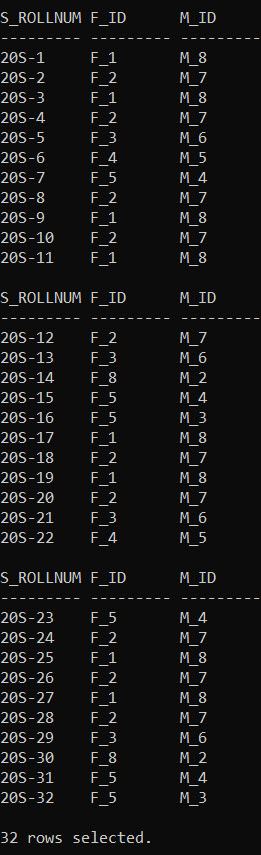
create view unique\_parents as select distinct f\_id,m\_id from student;

drop view siblings;

create view siblings as select s.s\_rollnumber,s.f\_id,s.m\_id from student s ,unique\_parents u where u.f\_id = s.f\_id and u.m\_id = s.m\_id;

select \* from siblings;

select r.class\_id,s.s\_rollnumber,std.s\_name,f.f\_name,m.m\_name from registration r, siblings s,mother m, father f, student std where s.s\_rollnumber = std.s\_rollnumber and s.F\_ID = f.f\_id and s.M\_ID = m.m\_id and s.s\_rollnumber = r.S\_ROLLNUMBER order by r.class\_id;



create view male1 as select count('M') male,r.class\_id from student s,registration r where s.s\_rollnumber = r.s\_rollnumber and s.s\_gender = 'M' group by class\_id;

create view female1 as select count('M') female,r.class\_id from student s,registration r where s.s\_rollnumber = r.s\_rollnumber and s.s\_gender = 'F' group by class\_id;

create view boht as select m.male,f.female,m.class\_id from male1 m,female1 f where m.class\_id = f.class\_id;

create view un\_male as

select class\_id from male1

minus

select class\_id from boht;

create view un\_female as

select class\_id from female1

minus

select class\_id from boht;

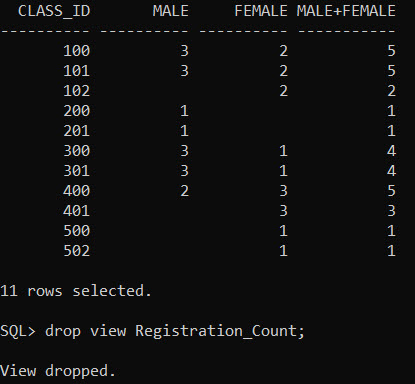
select b.class\_id,male,female,male+female from boht b

union

select class\_id,(select male from male1 m where un.class\_id=m.class\_id) ,NULL, (select male from male1 m where un.class\_id=m.class\_id) from un\_male un

union

select class\_id,NULL,(select female from female1 m where un.class\_id=m.class\_id)  ,(select female from female1 m where un.class\_id=m.class\_id) from un\_female un;



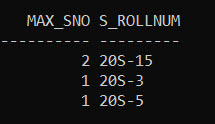
--Selects Max SNO for each Student

create view Registration\_Count as

select count(sno) Max\_Sno, s\_rollnumber from Registration\_History group by s\_rollnumber;

select \* from Registration\_count;

drop view Last\_Class;



--selects students who changed their class

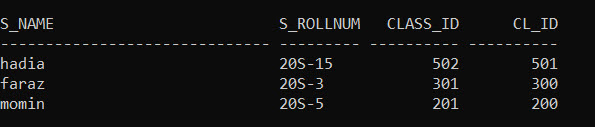
create view Class\_Change as

select s.s\_name, rgm.s\_rollnumber, rgm.class\_id, lc.cl\_id

from Registration\_Gap\_Month rgm, Last\_Class lc, Student s

where (rgm.s\_rollnumber = lc.s\_rollnumber and rgm.class\_id != lc.cl\_id and s.S\_ROLLNUMBER = lc.s\_rollnumber);

select \* from class\_change;



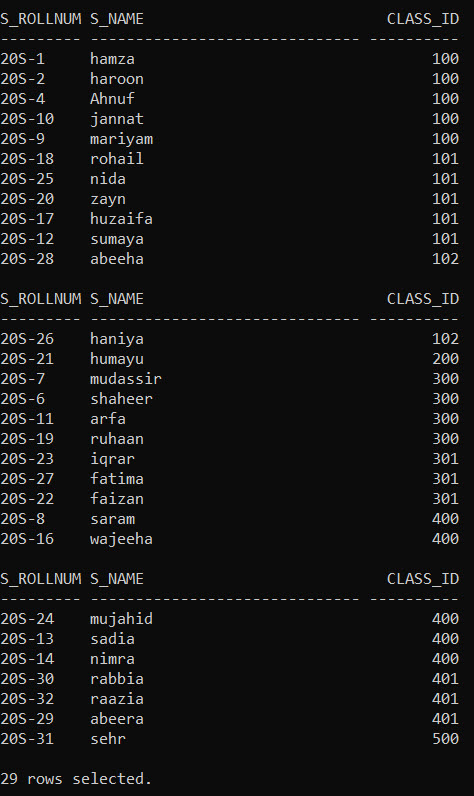
--Query 7

select na.s\_rollnumber,s.s\_name, r.class\_id

from New\_Admissions na, Registration r, student s

where na.s\_rollnumber = r.s\_rollnumber and na.s\_rollnumber = s.s\_rollnumber

order by r.class\_id;

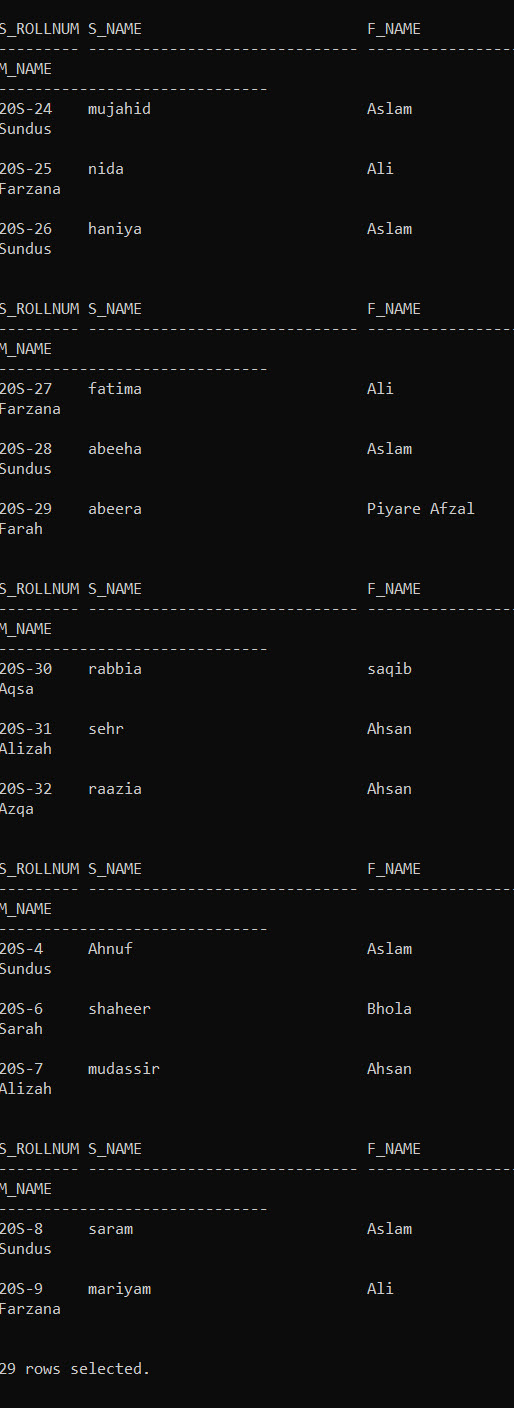


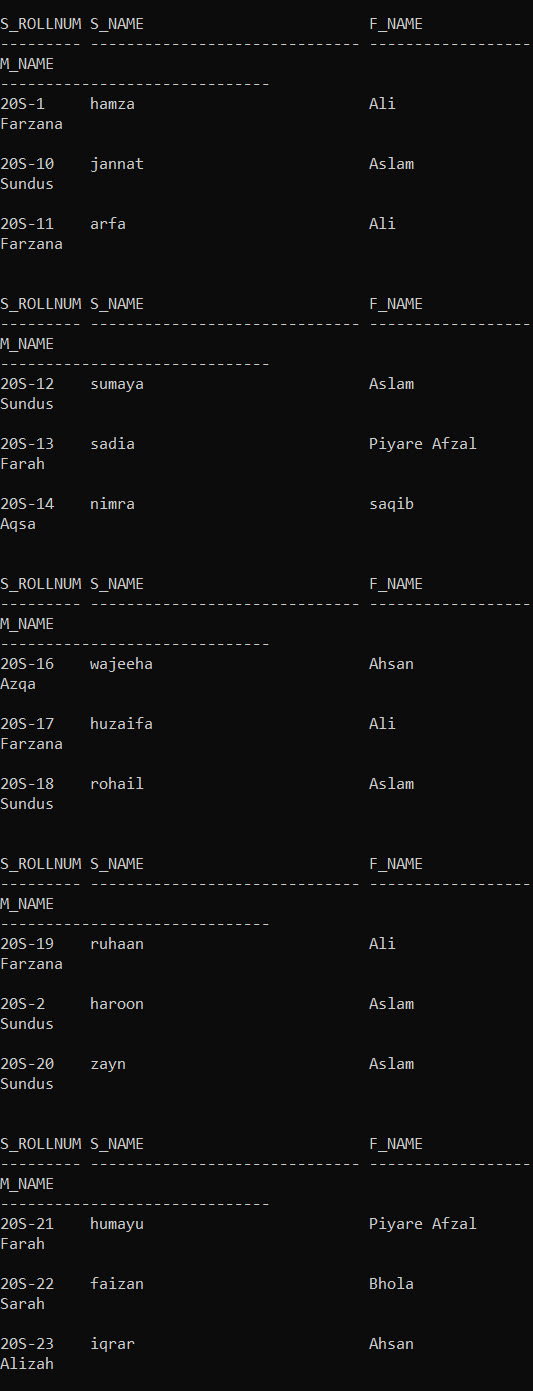
--Query 8

select na.s\_rollnumber,s.s\_name, f.f\_name,m.m\_name

from New\_Admissions na, student s, father f, mother m

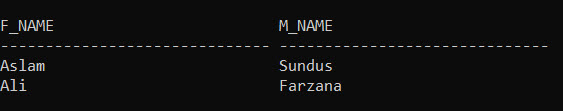
where  na.s\_rollnumber = s.s\_rollnumber and s.f\_id = f.f\_id and s.m\_id = m.m\_id;





--query 9

select distinct f.f\_name, m.m\_name from father f, mother m , student s,registration r where f.f\_id = s.f\_id and m.m\_id = s.m\_id and s.s\_rollnumber = r.S\_ROLLNUMBER and ((r.r\_date-s.dob)/365 >=2 and  (r.r\_date-s.dob)/365 <=5) ;



--Query 10 shows all students who changed their class

select s.s\_name, rh.s\_rollnumber, rh.cl\_id

from Registration\_History rh, Student s

where rh.s\_rollnumber = s.s\_rollnumber

union

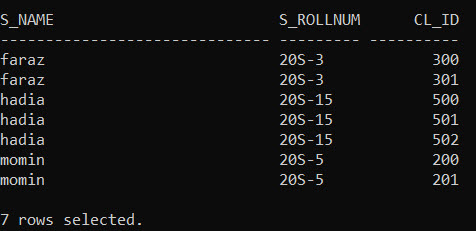
select s.s\_name, r.s\_rollnumber, r.class\_id

from Registration r, Registration\_History rh ,Student s

where r.s\_rollnumber = rh.s\_rollnumber

and r.class\_Id != rh.cl\_id

and r.s\_rollnumber = s.s\_rollnumber;



--Query 13

select rh.s\_rollnumber

from Registration\_History rh

where (MONTHS\_BETWEEN(sysdate,rh.R\_Date)>6) minus select r.s\_rollnumber from Registration r;

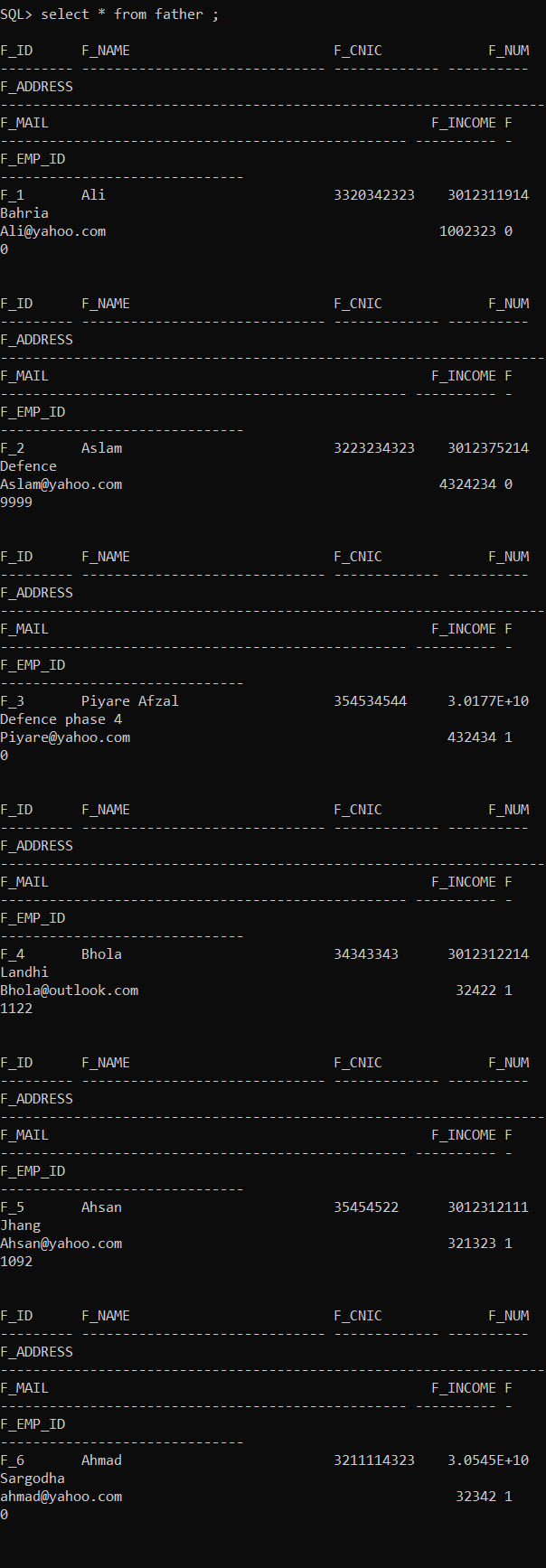
select rh.s\_rollnumber

from Registration\_History rh where (MONTHS\_BETWEEN(sysdate,rh.R\_Date)>6)

minus

select r.s\_rollnumber from Registration r;

select \* from father ;

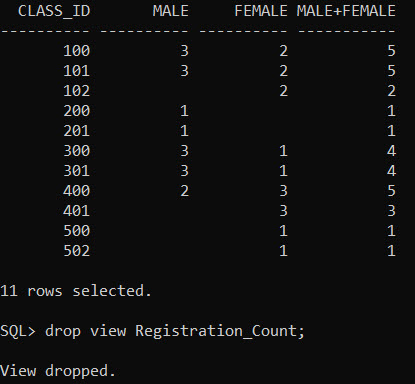


**Forms/Reports:**

At last we’ve created required forms and reports using PHP and Oracle.

We’ve created a report of a list of students grouped by classes along with add, search, delete, and edit student features and functionalities. In this form search can be performed using student name or id and it’ll fetch list of those students.

We’ve created a report of a list of classes with number of students per class and student count per gender. search can be performed using class name.



We’ve created a report of a list of all students who have been dormant for given number of months/years. Search can be performed using months or years.

We’ve created a report of all info on a given student (parents, guardian, siblings, class History). Meanwhile search can be performed using student name or id.

We’ve also created a report of all info on a given parent (all children, classes of each child, designated guardian). While search can be performed using parent name or id