SQL

TABLES

```
CREATE TABLE `feemgtsys`.`courses` (
  `Cid` CHAR(3) NOT NULL,
  `Cname` VARCHAR(128) NOT NULL,
  PRIMARY KEY (`Cid`));


CREATE TABLE `feemgtsys`.`studentinfo` (
  `Cid` CHAR(3) NOT NULL,
  `Year` INT(1) NOT NULL,
  `Sid` INT(4) NOT NULL,
  `Name` VARCHAR(32) NOT NULL,
  `Sex` CHAR(1) NOT NULL,
  `PhoneNo` VARCHAR(16) NOT NULL,
  `Email` VARCHAR(256) NOT NULL,
  PRIMARY KEY (`Cid`, `Year`, `Sid`),
  UNIQUE INDEX `Email_UNIQUE` (`Email` ASC));


CREATE TABLE `feemgtsys`.`refund` (
  `Cid` CHAR(3) NOT NULL,
  `Year` INT(1) NOT NULL,
  `Sid` INT(4) NOT NULL,
  `RefundAmt` DECIMAL(8,2) NOT NULL,
  PRIMARY KEY (`Cid`, `Year`, `Sid`));


CREATE TABLE `feemgtsys`.`studentfeeinfo` (
  `Cid` CHAR(3) NOT NULL,
  `Year` INT(1) NOT NULL,
  `Sid` INT(4) NOT NULL,
  `FeeStatus` INT(1) NOT NULL DEFAULT 0,
  `RefundStatus` INT(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (`Cid`, `Year`, `Sid`));
```

```
CREATE TABLE `feemgtsys`.`fee` (
  `Cid` CHAR(3) NOT NULL,
  `Year` INT(1) NOT NULL,
  `FeeAmt` DECIMAL(6,2) NOT NULL,
  PRIMARY KEY (`Cid`, `Year`));
```

```
CREATE TABLE `feemgtsys`.`login` (
  `username` VARCHAR(32) NOT NULL,
  `password` VARCHAR(128) NOT NULL,
  `privileges` CHAR(1) NOT NULL DEFAULT 'U',
  PRIMARY KEY (`username`));
```

REFERENTIAL INTEGRITY CONSTRAINTS

```
ALTER TABLE `feemgtsys`.`studentinfo`
ADD CONSTRAINT `studentinfo_fk_cid`
  FOREIGN KEY (Cid)
  REFERENCES `feemgtsys`.`courses` (Cid)
  ON DELETE CASCADE
  ON UPDATE CASCADE;


ALTER TABLE `feemgtsys`.`studentfeeinfo`
ADD INDEX `studentinfo_fk_Stcode_idx` (`Cid` ASC,`Year` ASC, `Sid`
ASC);
ALTER TABLE `feemgtsys`.`studentfeeinfo`
ADD CONSTRAINT `studentinfo_fk_Stcode`
  FOREIGN KEY (`Cid`, `Year` , `Sid`)
  REFERENCES `feemgtsys`.`studentinfo` (`Cid`, `Year` , `Sid`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;


ALTER TABLE `feemgtsys`.`refund`
ADD INDEX `refund_fk_StCode_idx` (`Cid` ASC, `Year` ASC, `Sid` ASC);
ALTER TABLE `feemgtsys`.`refund`
ADD CONSTRAINT `refund_fk_StCode`
  FOREIGN KEY (`Cid`, `Year` , `Sid`)
  REFERENCES `feemgtsys`.`studentfeeinfo` (`Cid`, `Year` , `Sid`)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE `feemgtsys`.`fee`
ADD CONSTRAINT `fee_fk_cid`
  FOREIGN KEY (Cid)
  REFERENCES `feemgtsys`.`courses` (Cid)
  ON DELETE CASCADE
  ON UPDATE CASCADE;
```

TRIGGERS

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.login_BEFORE_INSERT$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`login_BEFORE_INSERT` BEFORE INSERT ON `login` FOR EACH
ROW
BEGIN
	IF NEW.privileges!='U' AND NEW.privileges!='S' AND
NEW.privileges!='A'
	THEN
			SIGNAL SQLSTATE '45000'
			SET MESSAGE_TEXT = 'Invalid privileges';
	END IF;
END$$
DELIMITER ;
```

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.login_BEFORE_UPDATE$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`login_BEFORE_UPDATE` BEFORE UPDATE ON `login` FOR EACH
ROW
BEGIN
	IF NEW.privileges!='U' AND NEW.privileges!='S' AND
NEW.privileges!='A'
	THEN
			SIGNAL SQLSTATE '45000'
			SET MESSAGE_TEXT = 'Invalid privileges';
	END IF;
END$$
DELIMITER ;
```

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.studentinfo_BEFORE_UPDATE$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`studentinfo_BEFORE_UPDATE` BEFORE UPDATE ON `studentinfo`
FOR EACH ROW
BEGIN
     IF NEW.Sex!='M' AND NEW.Sex!='F' AND NEW.Sex!='O'
     THEN
               SIGNAL SQLSTATE '45000'
               SET MESSAGE_TEXT = 'Invalid values in Column "Sex"';
     END IF;
END$$
DELIMITER ;
```

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.studentinfo_BEFORE_INSERT$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`studentinfo_BEFORE_INSERT` BEFORE INSERT ON `studentinfo`
FOR EACH ROW
BEGIN
     IF NEW.Sex!='M' AND NEW.Sex!='F' AND NEW.Sex!='O'
     THEN
               SIGNAL SQLSTATE '45000'
               SET MESSAGE_TEXT = 'Invalid values in Column "Sex"';
     END IF;

END$$
DELIMITER ;
```

```
USE `feemgtsys`;
DELIMITER $$
DROP TRIGGER IF EXISTS feemgtsys.studentfeeinfo_BEFORE_INSERT$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`studentfeeinfo_BEFORE_INSERT` BEFORE INSERT ON `studentfeeinfo`
FOR EACH ROW
BEGIN
      IF NEW.RefundStatus!=0 AND NEW.RefundStatus!=1
      THEN
                  SIGNAL SQLSTATE '45000'
                  SET MESSAGE_TEXT = 'Invalid Refund Status!';
      ELSEIF NEW.FeeStatus!=0 AND NEW.FeeStatus!=1
    THEN
                  SIGNAL SQLSTATE '45000'
                  SET MESSAGE_TEXT = 'Invalid Fee Status!';
      ELSEIF NEW.FeeStatus=0 AND NEW.RefundStatus=1
    THEN
                  SIGNAL SQLSTATE '45000'
                  SET MESSAGE_TEXT = 'Fee cannot be refunded';
      END IF;
END$$
DELIMITER ;
```

```
USE `feemgtsys`;
DELIMITER $$
DROP TRIGGER IF EXISTS feemgtsys.studentfeeinfo_BEFORE_UPDATE$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`studentfeeinfo_BEFORE_UPDATE` BEFORE UPDATE ON `studentfeeinfo`
FOR EACH ROW
BEGIN
      IF NEW.RefundStatus!=0 AND NEW.RefundStatus!=1
      THEN
                  SIGNAL SQLSTATE '45000'
                  SET MESSAGE_TEXT = 'Invalid Refund Status!';
      ELSEIF NEW.FeeStatus!=0 AND NEW.FeeStatus!=1
    THEN
                  SIGNAL SQLSTATE '45000'
                  SET MESSAGE_TEXT = 'Invalid Fee Status!';
      ELSEIF NEW.FeeStatus=0 AND NEW.RefundStatus=1
    THEN
                  SIGNAL SQLSTATE '45000'
                  SET MESSAGE_TEXT = 'Fee cannot be refunded';
      END IF;
END$$
DELIMITER ;
```

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.studentinfo_AFTER_INSERT$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`studentinfo_AFTER_INSERT` AFTER INSERT ON `studentinfo`
FOR EACH ROW
BEGIN
     SET @COUNT=(SELECT COUNT(*) FROM studentfeeinfo WHERE
(cid=NEW.cid AND year=NEW.year AND sid=NEW.sid));
     IF @COUNT=0 THEN
        INSERT INTO studentfeeinfo
VALUES(NEW.cid,NEW.year,NEW.sid,0,0);
    ELSE
        UPDATE studentfeeinfo SET studentfeeinfo.cid=NEW.cid AND
studentinfo.year=NEW.year AND studentinfo.sid= NEW.sid
        WHERE (cid=NEW.cid AND year=NEW.year AND sid=NEW.sid);
    END IF;
END$$
DELIMITER ;
```

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.studentinfo_AFTER_UPDATE$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`studentinfo_AFTER_UPDATE` AFTER UPDATE ON `studentinfo`
FOR EACH ROW
BEGIN
      UPDATE studentfeeinfo SET studentfeeinfo.cid=NEW.cid AND
studentinfo.year=NEW.year AND studentinfo.sid= NEW.sid
      WHERE (cid=NEW.cid AND year=NEW.year AND sid=NEW.sid);
END$$
DELIMITER ;
```

```
USE `feemgtsys`;

DELIMITER $$

DROP TRIGGER IF EXISTS feemgtsys.refund_BEFORE_INSERT$$
USE `feemgtsys`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`feemgtsys`.`refund_BEFORE_INSERT` BEFORE INSERT ON `refund` FOR EACH
ROW
BEGIN
     SET @CHECK1=(SELECT COUNT(*) FROM studentfeeinfo WHERE
(cid=NEW.cid AND year=NEW.year AND sid=NEW.sid AND feeStatus=0));
    SET @CHECK2=(SELECT COUNT(*) FROM studentfeeinfo WHERE
(cid=NEW.cid AND year=NEW.year AND sid=NEW.sid AND refundStatus=0));

    IF @CHECK1!=0 THEN
           SIGNAL SQLSTATE '45000'
                 SET MESSAGE_TEXT = 'Invalid insertion! Fee has not
been submitted';
     ELSEIF @CHECK2!=0 THEN
           SIGNAL SQLSTATE '45000'
                 SET MESSAGE_TEXT = 'Invalid insertion! Fee has not
been refunded';
     END IF;
END$$
DELIMITER ;
```

VIEWS

```
USE `feemgtsys`;
CREATE  OR REPLACE VIEW `STUDENT` AS
     select cname, s.year, s.sid, s.name, s.sex, s.email, feeStatus
    from courses, studentinfo s, studentfeeinfo sf
    where (courses.cid = s.cid AND s.cid = sf.cid AND s.year= sf.year
AND s.sid AND sf.sid);;
```