Final Lab Exam Tasks:

## Functional Dependencies:

* Employee (Employee ID, Employee Name, Job, Age, Character ID, Student ID)
* Employee ID-> Employee Name, Job
* Employee Name-> Age Offices (Office ID, Office Name, Level ID, Employee ID, In charge ID)
* Office ID->Office Name In charge (In charge ID, In charge Name, Hire Date)
* In charge ID -> In charge Name, Hire Date Ministers (Minister ID, Minister Name, Hire Date)
* Minister ID ->Minister Name, Hire Date Characters (Character ID, Character Name, Role)
* Character ID -> Character Name
* Character Name ->Role Books (Book ID, Book Name, Author, Published By, Published Date)
* Book ID ->Book Name, Author , Published By, Published Date
* Book Name -> Author Levels (Level ID, Level Name, Minister Id)
* Level ID ->Level Name Student (Student ID, Student Name, Character ID, Year)
* Student ID -> Student Name ,Year
* Character ID -> Student Name Professors (Professor ID, Professor Name, Age, Character ID, Course ID)
* Professor ID -> Professor Name
* Professor Name◊ Age
* Examiners (Examiner ID, Examiner Name, Character ID, Level ID)
* Examiner ID-> Examiner Name Course (Course ID, Course Name, Nature Credit Hours)
* Course ID ->Course Name
* Course Name ->Nature, Credit Hours OWL (Course ID, Examiner ID, Slot, Nature)
* Course ID, Examiner ID -> Slot
* Course ID -> Examiner ID, Nature Course Nature (Course ID, Nature)
* Course ID -> Nature School (School ID, School Name, Current Head, Deputy Head, Region)
* School ID ->School Name
* School Name-> Current Head, Deputy Head, Region Grades (Student ID, Course ID, Grade, Year)
* Student ID, Course ID -> Grade
* Student ID->Year
* Enrollment (Student ID, Course ID)
* Appearance (Book ID, Character ID)

# ERD:

The diagram was designed by using diagrams.net which is an online utility to design flowcharts.

Diagram

Description automatically generated

# Database Script(with constraints):

CREATE TABLE WhizardingSchool(

    SchoolID NUMBER(5) NOT NULL,

    SchoolName VARCHAR2(15) NOT NULL,

    CurrentHead VARCHAR2(15) NOT NULL,

    DeputyHead VARCHAR2(15) NOT NULL,

    Region VARCHAR2(13) NOT NULL,

CONSTRAINT PkSchool PRIMARY KEY(SchoolID)

)

/

CREATE TABLE Books(

    ISBN NUMBER(5) NOT NULL,

    Author VARCHAR2(15) NOT NULL,

    PublishedBy VARCHAR2(15) NOT NULL,

    PublishedDate Date NOT NULL,

CONSTRAINT PkBook PRIMARY KEY(ISBN)

)

/

CREATE TABLE Characters(

    CharID NUMBER(5) NOT NULL,

    CharName VARCHAR2(25) Not Null,

CONSTRAINT PkCharacter PRIMARY KEY(CharID)

)

/

Create table Enrollment

(StdID Number Not Null,

CourseID Number Not Null,

Constraint FEnKStudent Foreign Key (StdID) References Student(StdID),

Constraint FEnKCourse Foreign Key (CourseID) References Course(CourseID)

)

/

CREATE TABLE Appearance(

    ISBN NUMBER(5) NOT NULL,

    CharID NUMBER(5) NOT NULL,

    CONSTRAINT ISBN  FOREIGN KEY (ISBN) REFERENCES Books(ISBN),

    CONSTRAINT CharID FOREIGN KEY (CharID) REFERENCES Characters(CharID)

)

/

CREATE TABLE Student(

    StdId NUMBER(5) NOT NULL,

    StdName VARCHAR2(15) NOT NULL,

    CharID NUMBER(15) NOT NULL,

    SchoolID NUMBER(15) NOT NULL,

    Year NUMBER NOT NULL,

CONSTRAINT PkStudent PRIMARY KEY(StdId),

CONSTRAINT CharIDFk FOREIGN KEY (CharID) REFERENCES Characters(CharID),

CONSTRAINT SchoolIDFk FOREIGN KEY (SchoolID) REFERENCES WhizardingSchool(SchoolID)

)

/

CREATE TABLE Course(

    CourseId NUMBER(5) NOT NULL,

    CourseName VARCHAR2(15) NOT NULL,

    CDHr NUMBER(15) NOT NULL,

CONSTRAINT PkCourse PRIMARY KEY(CourseId)

)

/

CREATE TABLE Grades(

    StdId NUMBER(5) NOT NULL,

    CourseId NUMBER(5) NOT NULL,

    Grade VARCHAR2(12) NOT NULL,

    SchoolID NUMBER(15) NOT NULL,

    Year NUMBER NOT NULL,

CONSTRAINT StdIdFk FOREIGN KEY (StdId) REFERENCES Student(StdId),

CONSTRAINT CourseIDFk FOREIGN KEY (CourseId) REFERENCES Course(CourseId),

CONSTRAINT GradeObt CHECK(Grade IN('O','E','P','A','D','T'))

)

/

CREATE TABLE Student(

    StdId NUMBER(5) NOT NULL,

    StdName VARCHAR2(15) NOT NULL,

    CharID NUMBER(15) NOT NULL,

    SchoolID NUMBER(15) NOT NULL,

    Year NUMBER NOT NULL,

CONSTRAINT PkStudent PRIMARY KEY(StdId),

CONSTRAINT CharIDFk FOREIGN KEY (CharID) REFERENCES Characters(CharID),

CONSTRAINT SchoolIDFk FOREIGN KEY (SchoolID) REFERENCES WhizardingSchool(SchoolID)

)

/

CREATE TABLE Professor(

    ProfId NUMBER(5) NOT NULL,

    ProfName VARCHAR2(15) NOT NULL,

    CharID NUMBER(15) NOT NULL,

    CourseId NUMBER(5) NOT NULL,

CONSTRAINT ProfessorID PRIMARY KEY(ProfId),

CONSTRAINT ProfCharIDFk FOREIGN KEY (CharID) REFERENCES Characters(CharID),

CONSTRAINT ProfCourseIDFk FOREIGN KEY (CourseId) REFERENCES Course(CourseId)

)

/

CREATE TABLE Minister(

    MinisterId NUMBER(5) NOT NULL,

    MinisterName VARCHAR2(15) NOT NULL,

    Hiredate Date NOT NULL,

CONSTRAINT MinisterID PRIMARY KEY(MinisterId)

)

/

CREATE TABLE LEVAL(

    LevalId NUMBER(5) NOT NULL,

    LevalName VARCHAR2(15) NOT NULL,

    MinisterId NUMBER(5) NOT NULL,

CONSTRAINT LevalId PRIMARY KEY(LevalId),

CONSTRAINT MinisterIDFk FOREIGN KEY (MinisterId) REFERENCES Minister(MinisterId)

)

/

CREATE TABLE OfficeIncharge(

    InchargeId NUMBER(5) NOT NULL,

    InchargeName VARCHAR2(15) NOT NULL,

    Hiredate Date NOT NULL,

CONSTRAINT IChagrgeID PRIMARY KEY(InchargeId)

)

/

CREATE TABLE EmployeeMinistry(

    EmployeeID NUMBER(5) NOT NULL,

    EmployeeName VARCHAR2(15) NOT NULL,

    Age NUMBER(15) NOT NULL,

    Job NUMBER(15) NOT NULL,

    CharID NUMBER(15) NOT NULL,

    StdId NUMBER(5) NOT NULL,

CONSTRAINT PkEmployee PRIMARY KEY(EmployeeID),

CONSTRAINT EmpCharIDFk FOREIGN KEY (CharID) REFERENCES Characters(CharID),

CONSTRAINT EmpStdIDFk FOREIGN KEY (StdId) REFERENCES Student(StdId)

)

/

CREATE TABLE Office(

    OfficeID NUMBER(5) NOT NULL,

    OfficeName VARCHAR2(15) NOT NULL,

    EmployeeID NUMBER(5) NOT NULL,

    LevalId NUMBER(5) NOT NULL,

    InchargeId NUMBER(5) NOT NULL,

CONSTRAINT PkOffice PRIMARY KEY(OfficeID),

CONSTRAINT OfficeLeval FOREIGN KEY (LevalID) REFERENCES LEVAL(LevalID),

CONSTRAINT InchargeOffice FOREIGN KEY (InchargeId) REFERENCES OfficeIncharge(InchargeId)

)

/

CREATE TABLE Examminer(

    ExamminerID NUMBER(5) NOT NULL,

    ExamminerName VARCHAR2(15) NOT NULL,

    CharID NUMBER(5) NOT NULL,

    LevalId NUMBER(5) NOT NULL,

CONSTRAINT PkExamminer PRIMARY KEY(ExamminerID),

CONSTRAINT ExamminerLeval FOREIGN KEY (LevalID) REFERENCES LEVAL(LevalID),

CONSTRAINT Examminerchar FOREIGN KEY (CharID) REFERENCES Characters(CharID)

)

/

Create Table OWLExam(

    ExamminerID Number Not Null,

    CourseID Number Not Null,

    Slot Number(2) Not Null,

    Nature Varchar2(20) Not Null,

Constraint FOKExaminer Foreign Key (ExamminerID) References Examminer(ExamminerID),

Constraint FOKCourse Foreign Key (CourseID) References Course(CourseID),

Constraint Slotted Check ((Slot Between 1 And 19)),

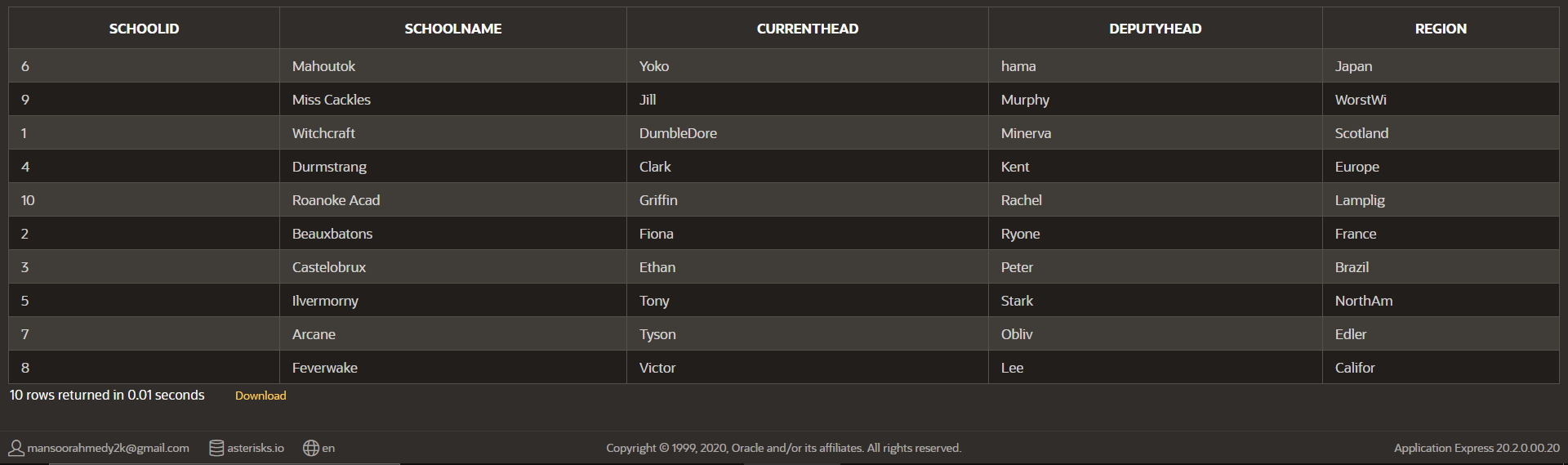
Constraint OwlsExamNature Check ((Nature In('Theory','Practical','Theory&Practical') ))

)

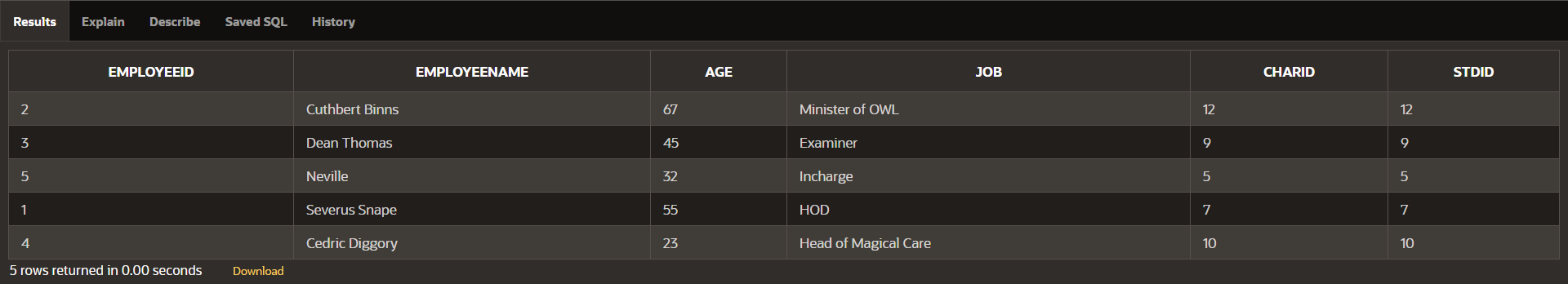
/

# Inserted Data Set:

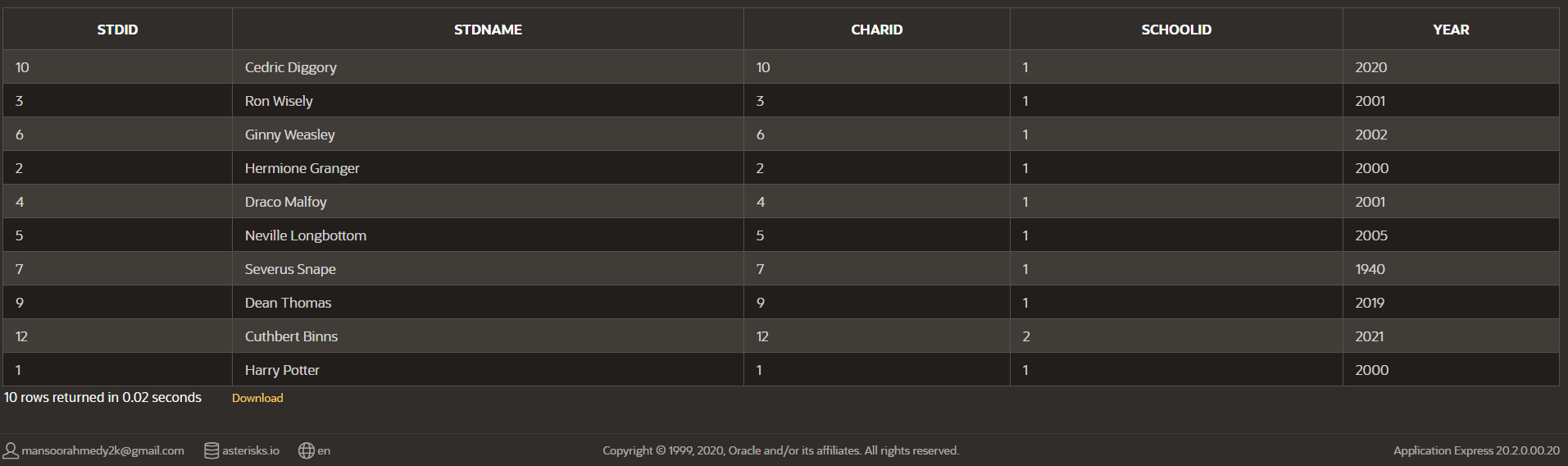
Wizarding School:



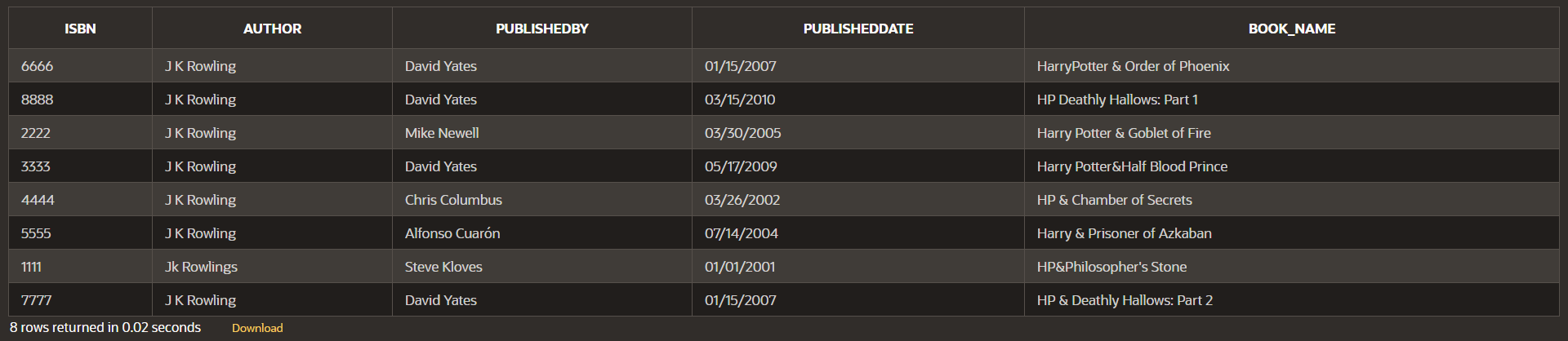
EmployeeMInistry:



Students:



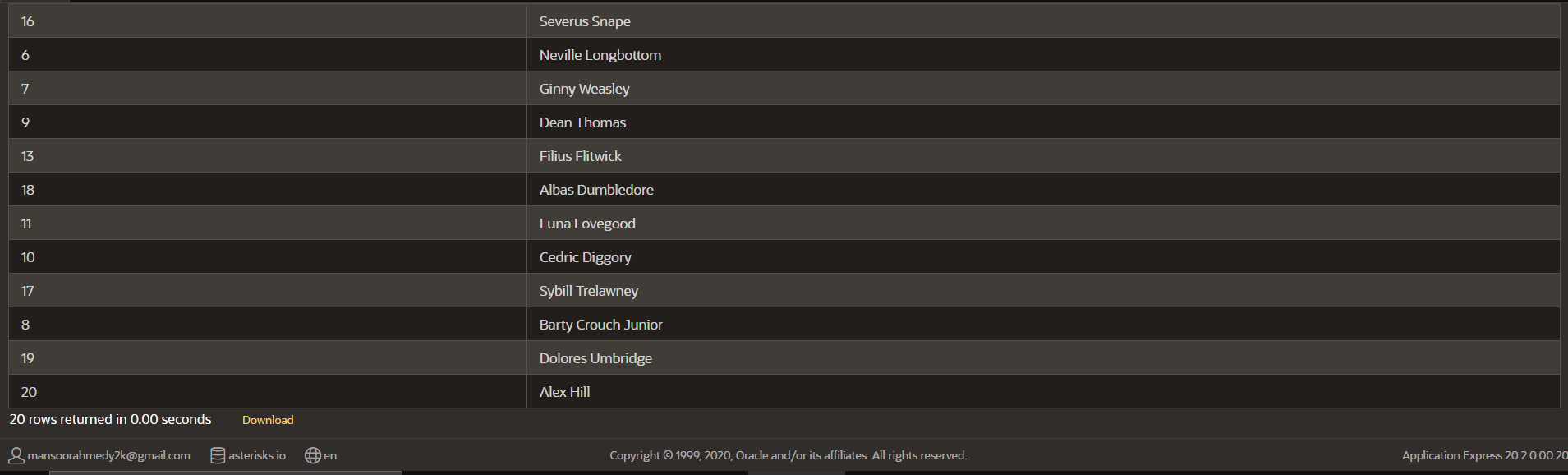
Book:



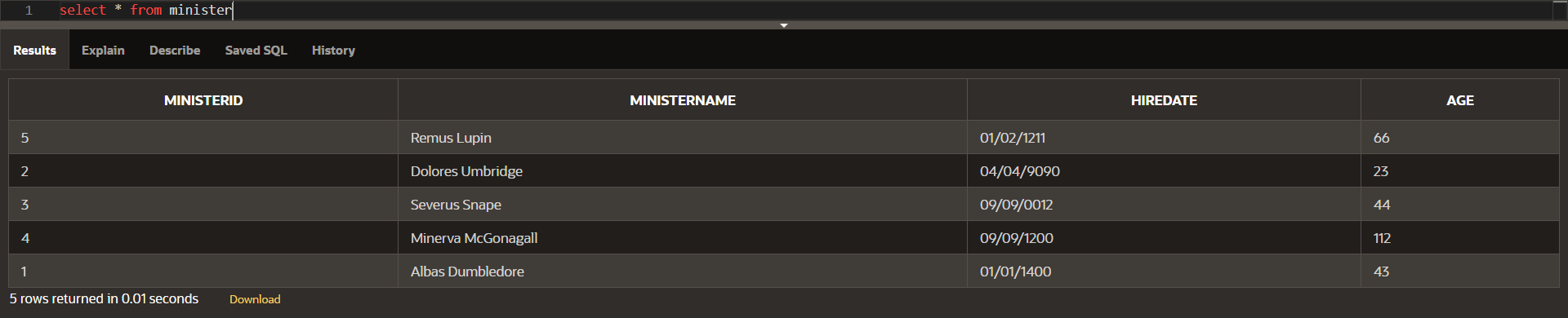
Appearance:



Characters:



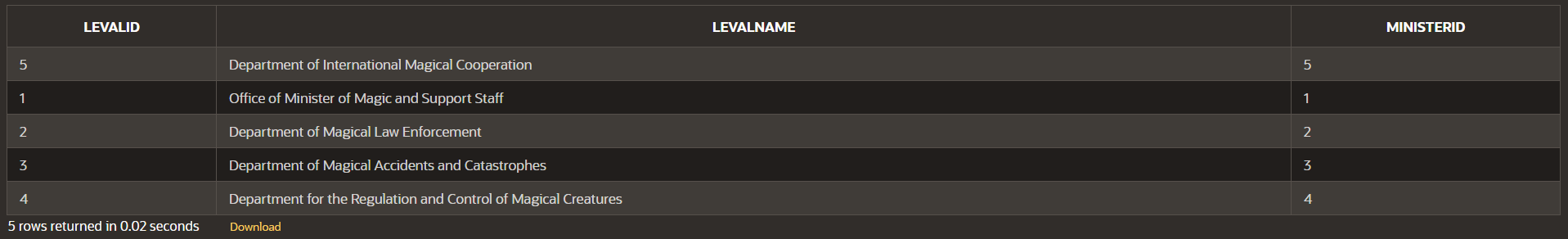
Minister:



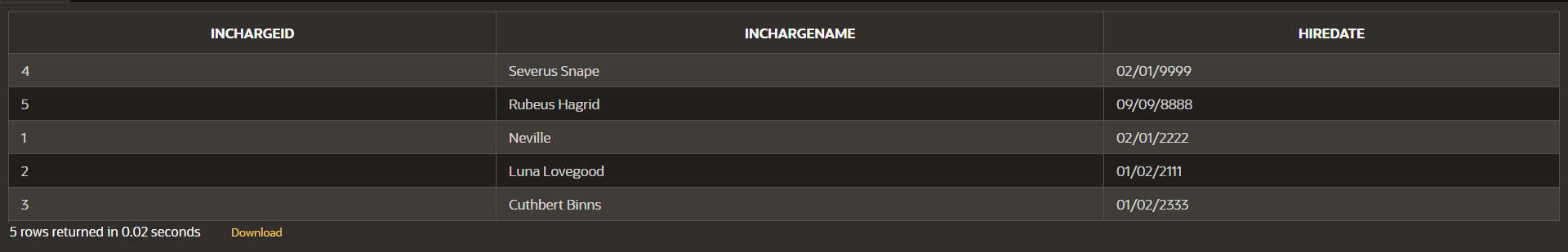
Office:



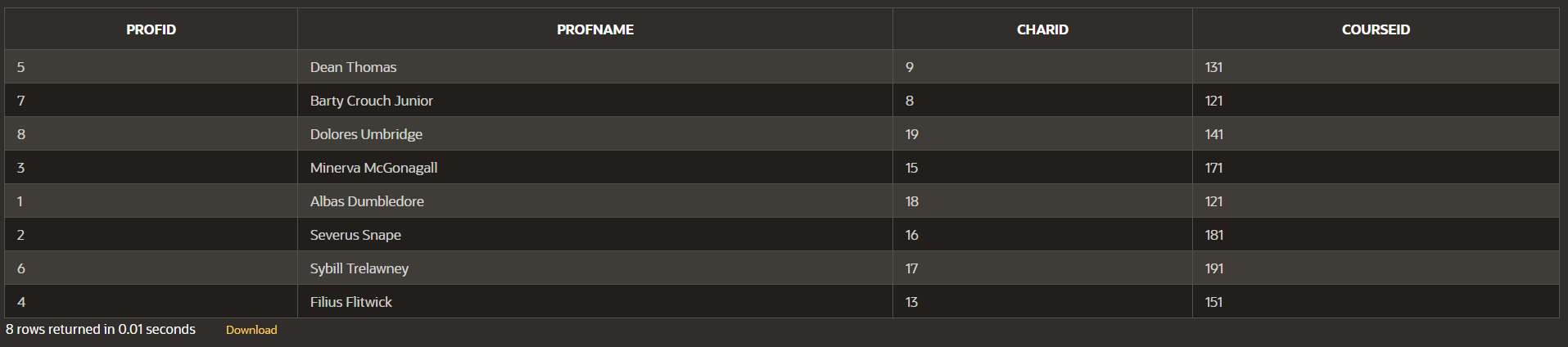
Level:



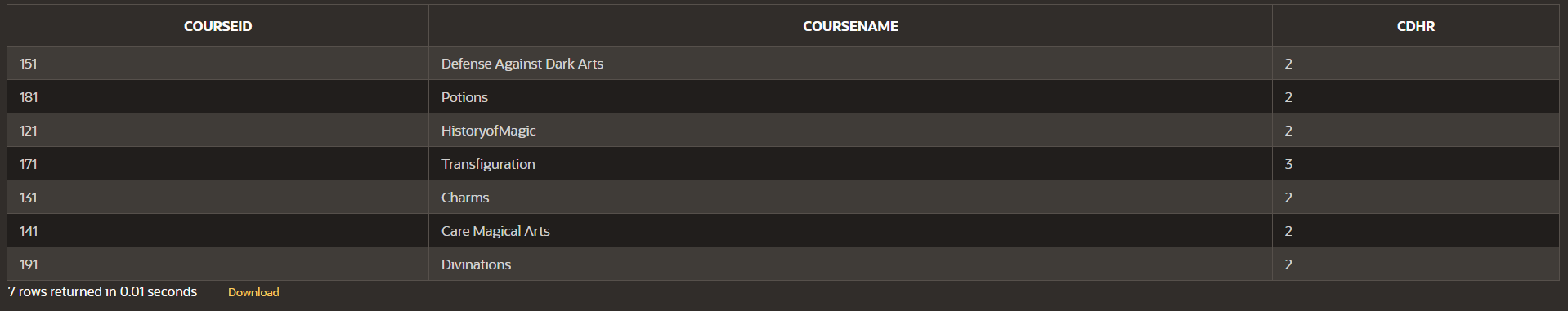
Office in charge:



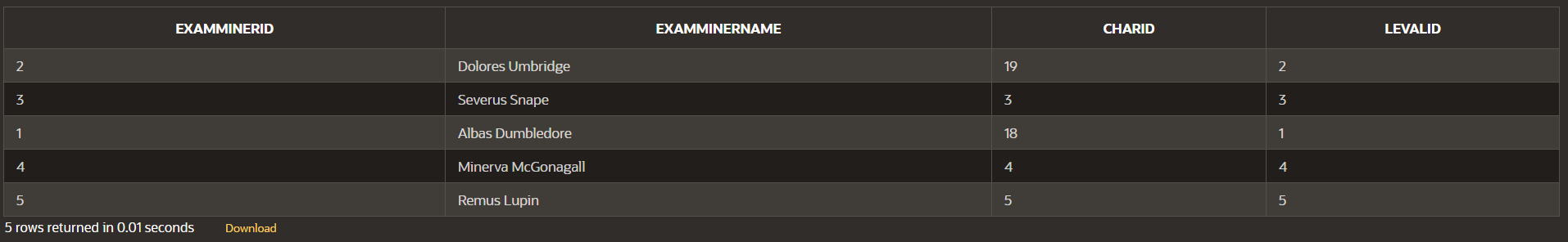
Professor:



Course:



Examiner:



# Queries:

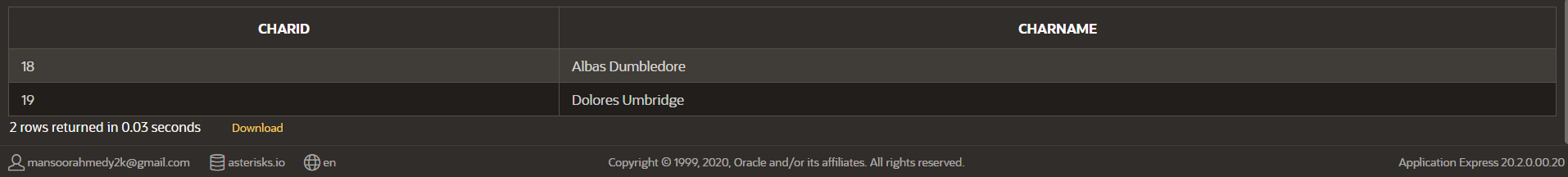
## Query-A:

SELECT CHARID,CHARNAME

FROM CHARACTERS

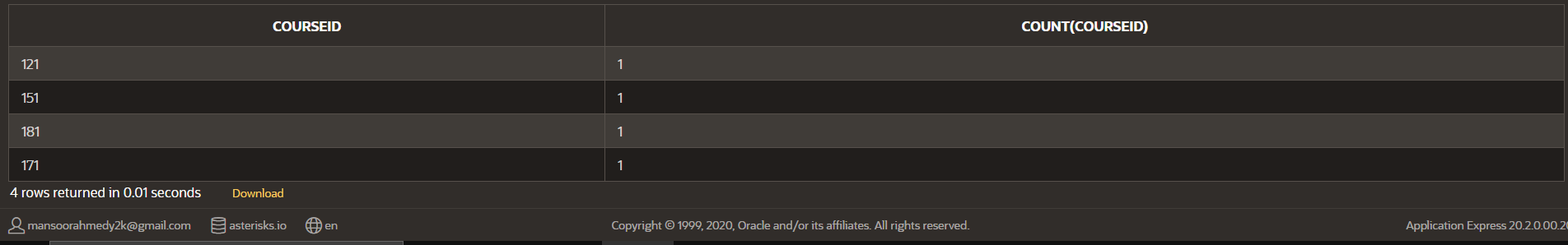
WHERE CHARID=ANY(SELECT CHARID FROM APPEARANCE WHERE ISBN = 1111 and CHARID<>ALL(SELECT CHARID FROM APPEARANCE WHERE

ISBN IN(2222,3333,4444,5555,6666,7777,8888)));



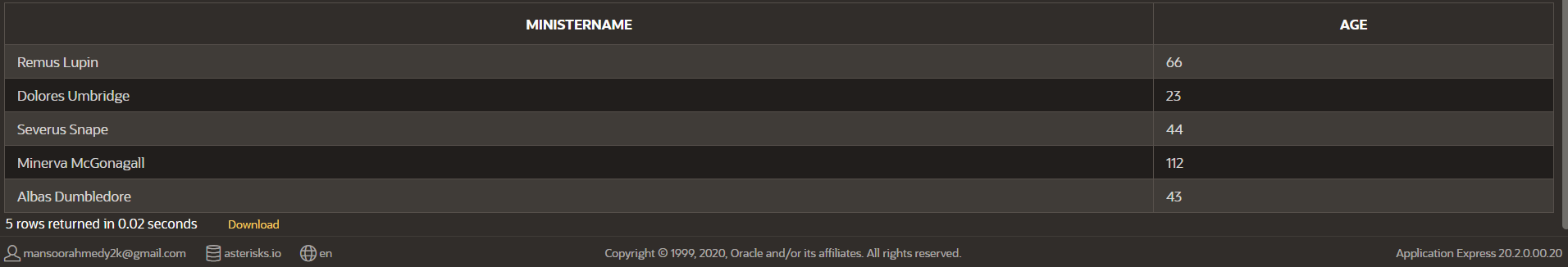
## Query-C:

SELECT COURSEID, COUNT(COURSEID) FROM ENROLLMENT GROUP BY COURSEID ORDER BY COUNT(COURSEID) ASC FETCH FIRST 1 ROW WITH TIES;



## Query-D:

SELECT ministername,age from minister



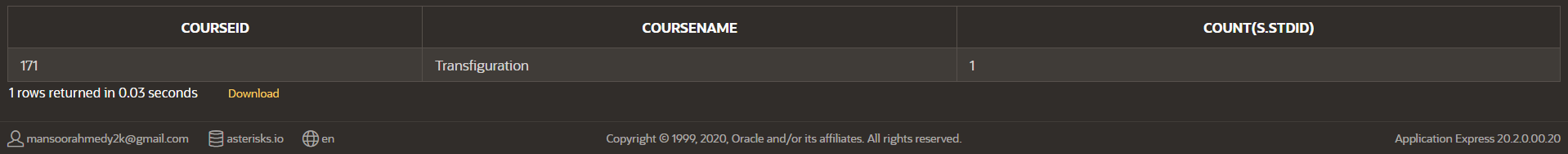
## Query-E:

SELECT O.COURSEID, C.COURSENAME, COUNT(S.STDID) FROM OWLExam O INNER JOIN COURSE C ON O.COURSEID=C.COURSEID

INNER JOIN grades G ON C.COURSEID=G.COURSEID INNER JOIN STUDENT S

ON G.STDID=S.STDID WHERE G.GRADE='P' and G.YEAR BETWEEN 2000 AND 2021

GROUP BY O.COURSEID, C.COURSENAME ORDER BY COUNT(O.COURSEID);



## Query-F

SELECT O.OFFICENAME,(SELECT E.EMPLOYEENAME FROM EMPLOYEEMINISTRY E WHERE

O.EMPLOYEEID=E.EMPLOYEEID) AS "EMPNAME",(SELECT L.LEVALNAME FROM LEVAL L WHERE

O.LEVALID=L.LEVALID) AS "LEVELNAME", COUNT(EMPLOYEEID) FROM OFFICE O

HAVING COUNT(O.EMPLOYEEID)=1 GROUP BY O.OFFICENAME,O.EMPLOYEEID, O.LEVALID;



## Query-G:

SELECT E.EMPLOYEENAME, COUNT(E.EMPLOYEENAME)

FROM EMPLOYEEMINISTRY E

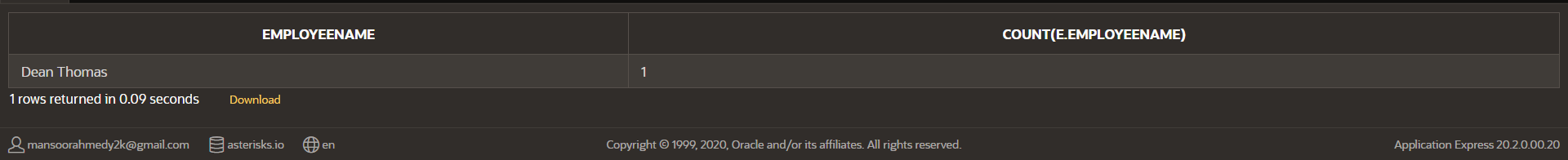
INNER JOIN OFFICE O ON E.EMPLOYEEID=O.EMPLOYEEID

INNER JOIN LEVAL L ON O.LEVALID=L.LEVALID

INNER JOIN STUDENT S ON E.STDID=S.STDID

INNER JOIN GRADES G ON E.STDID=G.STDID where L.LEVALID=3

AND G.GRADE='P' GROUP BY EMPLOYEENAME ORDER BY COUNT(EMPLOYEENAME);

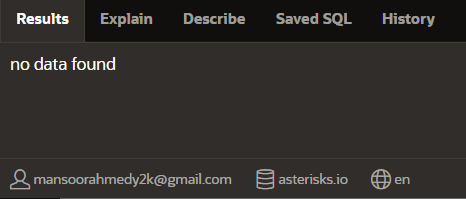


## Query-H:

SELECT L.LEVALNAME FROM EMPLOYEEMINISTRY E INNER JOIN OFFICE O

ON O.EMPLOYEEID=E.EMPLOYEEID INNER JOIN LEVAL L ON O.LEVALID=L.LEVALID GROUP BY

L.LEVALNAME HAVING COUNT(\*)>1;



# Advance Queries:

Joins

select G.grade,e.examminername from EMPLOYEEMINISTRY e

inner join office o ON e.employeeid=o.employeeid

inner join leval l ON l.levalid=o.levalid

inner join examminer e ON e.levalid=l.levalid

inner join OWLExam OW On e.examminerid=OW.examminerid

inner join Course C ON OW.courseid=C.courseiD

inner join grades G ON C.courseID=G.courseID ;

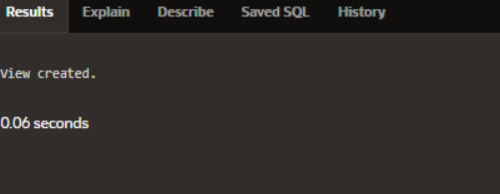


View

CREATE OR REPLACE VIEW MSTDETAILS AS

(SELECT L.LEVALNAME,O.OFFICENAME,E.EMPLOYEENAME FROM LEVAL L INNER JOIN OFFICE O ON

L.LEVALID=O.LEVALID INNER JOIN EM EMPLOYEEMINISTRY ON O.EMPLOYEEID=E.EMPLOYEEID);



## Application:

User:asterisks.io

PASS: mechanical

Email:mansoorahmedy2k@gmail.com

