**PE 1**

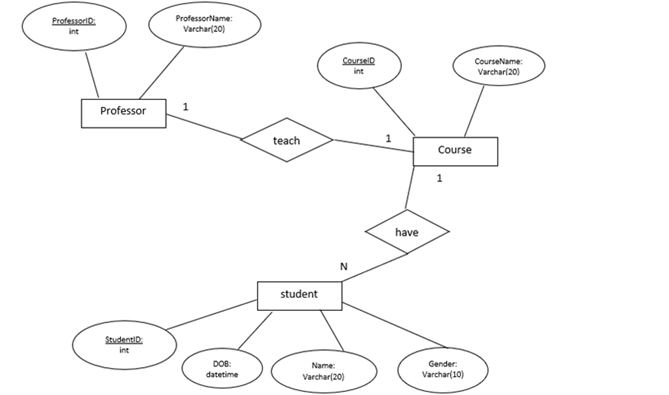
**Question 1:**

Create one database and then write SQL statements to create all tables derived from the ERD given in Picture 1.1. Those tables have to locate under the database that you have just created with appropriate attributes, primary keys and foreign keys.

NOTE that when creating the SQL commands as request, you MUST keep the name of tables, relationship and attributes and data type of attributes as SAME as given in the above ERD.

Attributes have written with underline are Primary Key of each entity.

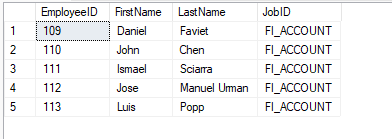
When submitting the responses for this question, submit only SQL statements for creating tables with corresponding keys and foreign keys. Do not use “create database” or “use database\_name” statements in your submission.



Picture 1.1

**Question 2:**

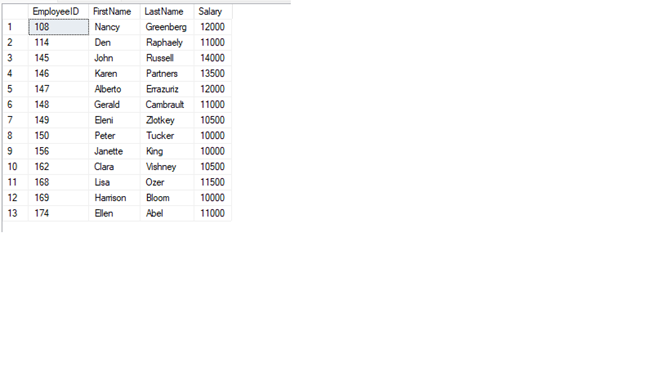
Write a query to display all employees that have JobID is FI\_ACCOUNT as following:



Picture 2.1

**Question 3:**

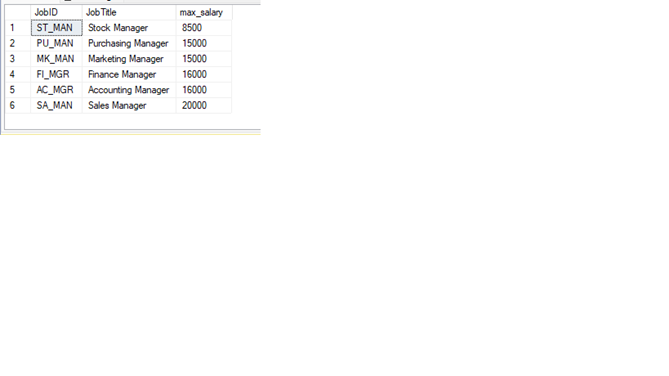
Write a query to display all employees that are Salary between 9000 and 15000 as following:



Picture 3.1

**Question 4:**

Write a query to select JobID, JobTitle, max\_salary of all ‘Manager’ ; display the results ascending order of max\_salary then in descending order of JobID for the jobs having the same max\_salary. Note that a job is called ‘Manager’ job if there is the word ‘Manager’ in its JobTitle.



Picture 4.1

**Question 5:**

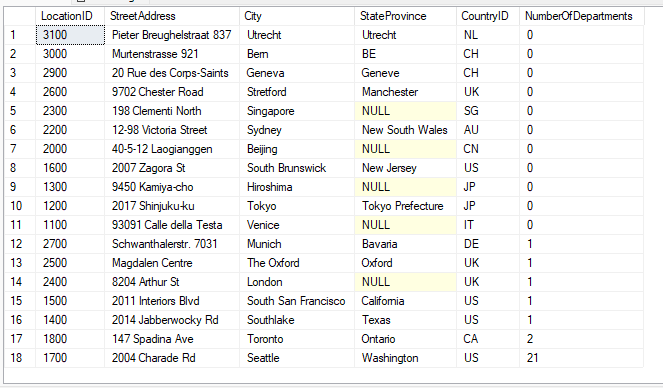
Write a query to select EmployeeID, FirstName, LastName, JobTitle, DepartmentName, year HireDate of all employees has year of HireDate is 2004.



Picture 5.1

**Question 6:**

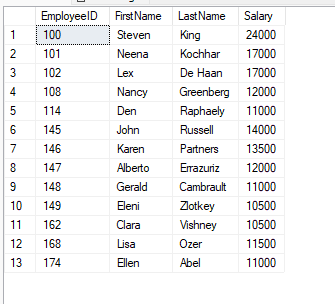
Write a query to display LocationID, StreetAddress, City, StateProvince, CountryID, NumberOfDepartment where NumberOfDepertment is the number of departments in each location; display the results in ascending order of NumberOfDepartment then in descending order of LocationID for locations having the same value of NumberOfDepartments as following:



Picture 6.1

**Question 7:**

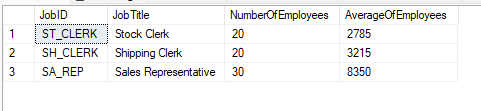
Write a query to display all employees has Salary greater than Salary of ‘Janette King’



Picture 7.1

**Question 8:**

Write a query to display JobID, JobTitle, NumberOfEmployees, AverageOfSalary of the jobs having the number of employees greater than or equal to 20 and display the results in ascending order of AverageOfSalary



Picture 8.1

**Question 9:**

Create a stored procedure named pr1 to calculate the number of departments in a given department where @departmentID varchar(10) is an input paramater and @numberOfEmployees int is an output parameter of the procedure.

For example, when we execute the procedure pr1 using the following statements, the result should be as in the following figure:

DECLARE @num int;

exec PR1 60, @num output;

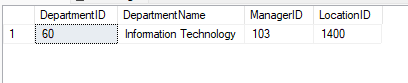
SELECT @num AS NumberOfEmployees



Picture 9.1

**Question 10:**

Update Name of ‘IT’ Department to ‘Information Technology’, and write a query to display IT Department as following:



Picture 10.1

SOLUTION

Q1: CREATE DATABASE PE1\_FALL\_2022

USE PE1\_FALL\_2022

CREATE TABLE Student

(

StudentID int PRIMARY KEY,

DOB datetime,

StudentName Varchar (20),

Gender Varchar (10),

CourseID int

)

CREATE TABLE Course

(

CourseID int PRIMARY KEY,

CourseName Varchar (20),

LecturerID int

)

CREATE TABLE Lecturer

(

LecturerID int PRIMARY KEY,

LecturerName Varchar (20),

)

ALTER TABLE Student

ADD CONSTRAINT Fore1 FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

ALTER TABLE Course

ADD CONSTRAINT Fore2 FOREIGN KEY (LecturerID) REFERENCES Lecturer(LecturerID)

Q2:

SELECT e.EmployeeID, e.FirstName, e.LastName, e.JobID

FROM Employees e

WHERE e.JobID='FI\_ACCOUNT'

Q3:

SELECT e.EmployeeID, e.FirstName, e.LastName, e.Salary

FROM Employees e

WHERE e.Salary BETWEEN 10000 AND 15000

Q4:

SELECT j.JobID,j.JobTitle,j.max\_salary

FROM Jobs j

WHERE j.jobTitle LIKE '%Manager%'

ORDER BY j.max\_salary asc, j.JobID desc

Q5:

SELECT e.EmployeeID, e.FirstName,e.LastName, j.JobTitle, d.DepartmentName,year(e.HireDate) as yearHireDate

FROM Jobs j JOIN Employees e ON j.JobID=e.JobID

JOIN Departments d ON e.DepartmentID=d.DepartmentID

WHERE year(e.HireDate)=2004

Q6:

select l.LocationID, l.StreetAddress, l.City, l.StateProvince, l.CountryID, count(d.LocationID) as NumberOfDepartments

from Departments d

right join Locations l

on d.LocationID = l.LocationID

group by l.LocationID, l.StreetAddress, l.City, l.StateProvince, l.CountryID

order by NumberOfDepartments ASC, l.LocationID DESC

Q7:

SELECT e.EmployeeID,e.FirstName,e.LastName,e.Salary

FROM Employees e

WHERE e.Salary>

(SELECT e.Salary

FROM Employees e

WHERE e.FirstName='Janette' AND e.LastName='King')

Q8:

select e.JobID,j.JobTitle ,count(e.JobID) as NumberOfEmployees,AVG(e.Salary) As AverageOfEmployees

from Employees e, Jobs j

where j.JobID = e.JobID

group by e.JobID,j.JobTitle

having count(e.JobID) >= 20

Order by AVG(e.Salary) asc

Q9:

CREATE OR ALTER PROC PR1 @departmentID int, @numberOfEmployees int output

AS

BEGIN

DECLARE @num int

SELECT @num = COUNT(\*) FROM Employees e WHERE e.DepartmentID = @departmentID

SET @numberOfEmployees = @num

END

DECLARE @num int;

exec PR1 60, @num output;

SELECT @num AS NumberOfEmployees

Q10:

UPDATE Departments

SET DepartmentName='Information Technology'

WHERE DepartmentName='IT'

SELECT \*

FROM Departments

WHERE DepartmentName='Information Technology'

PE 2

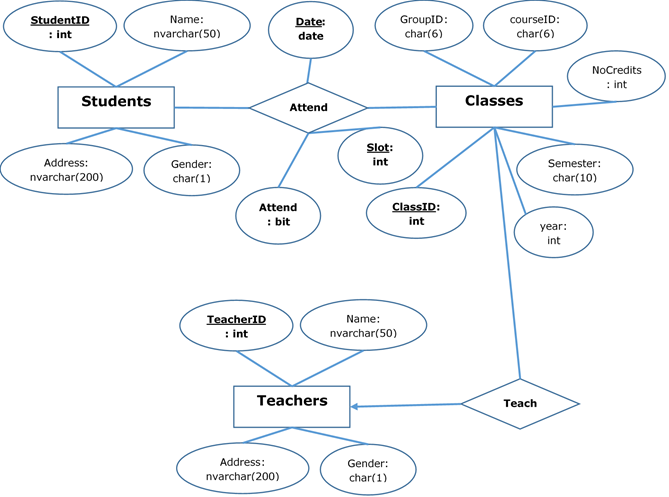
**Question 1:**

Create one database and then write SQL statements to create all tables derived from the ERD given in Picture 1.1. Those tables have to locate under the database that you have just created with appropriate attributes, primary keys and foreign keys.

NOTE that when creating the SQL commands as request, you MUST keep the name of tables, relationship and attributes and data type of attributes as SAME as given in the above ERD.

Attributes have written with underline are Primary Key of each entity.

When submitting the responses for this question, submit only SQL statements for creating tables with corresponding keys and foreign keys. Do not use “create database” or “use database\_name” statements in your submission.



Picture 1.1

create table Teachers(

TeacherID int primary key,

Name nvarchar(50),

Address nvarchar(200),

Gender char(1)

)

create table Classes(

ClassID int Primary Key,

GroupID char(6),

coueseID char(6),

NoCredits int,

Semester char(10),

year int,

TeacherID int

constraint fk\_Classes\_Teachers FOREIGN KEY(TeacherID) references Teachers(TeacherID)

)

create table Students(

StudentID int Primary Key,

Name nvarchar(50),

Address nvarchar(200),

Gender char(1),

)

create table Attend(

Date date,

Slot int,

Attend bit,

StudentID int,

ClassID int,

primary key(Date, Slot, StudentID, ClassID),

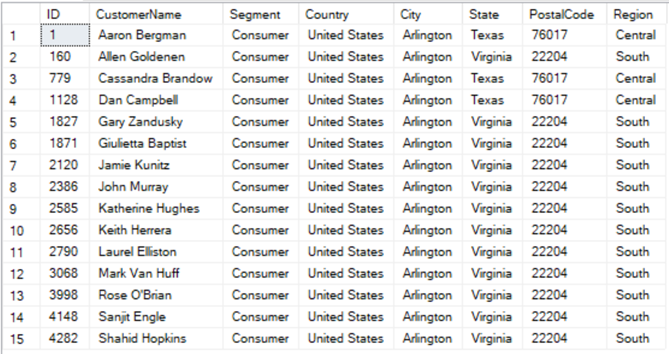
constraint fk\_Attend\_Students FOREIGN KEY(StudentID) references Students(StudentID),

constraint fk\_Attend\_Classes FOREIGN KEY(ClassID) references Classes(ClassID)

)

**Question 2:**

Write a query to display all customers who are ‘Consumer’ and are from Arlington city as follows:



Picture 2.1

**Question 3:**

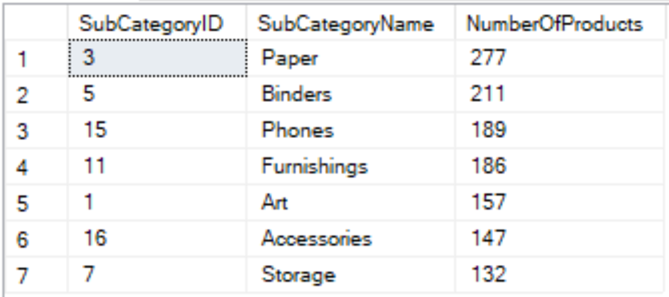
Write a query to display all customers having CustomerName starting with B and placed orders in December 2017. Display the result by descending order of Segment and then by ascending order of CustomerName.



Picture 3.1

**Question 4:**

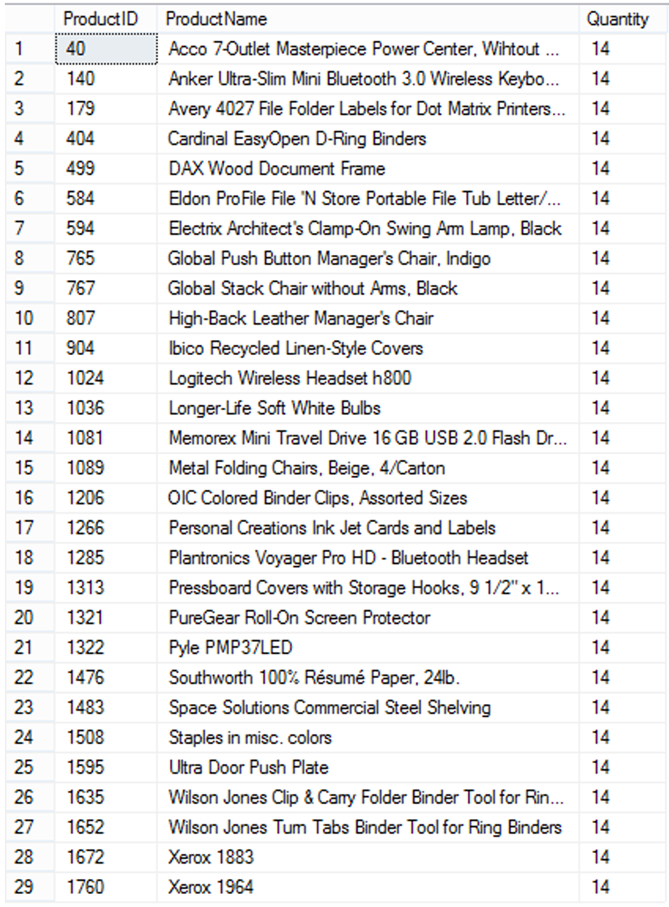
Write a query to display SubCategoryID, SubCategoryName and the corresponding number of products (NumberOfProducts) in each sub-category having the number of products greater than 100, by descending order of NumberOfProducts.



Picture 4.1

**Question 5:**

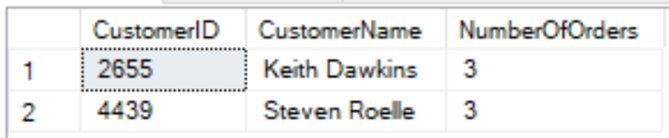
Write a query to display ProductID, ProductName, Quantity of all products which have the highest Quantity in one order.



Picture 5.1

**Question 6:**

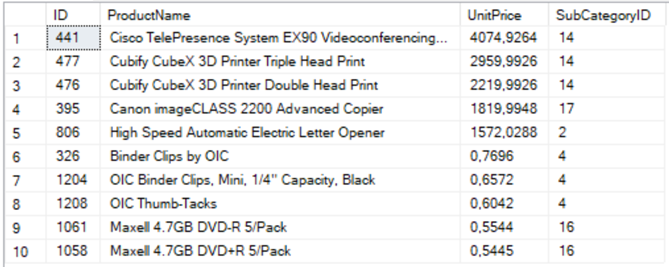
Write a query to display CustomerID, CustomerName and the number of orders (NumberOfOrders) of customers who have the highest number of orders.



Picture 6.1

**Question 7:**

Display 5 products with the highest unit prices and 5 products with the smallest unit prices as follows:



Picture 7.1

**Question 8:**

Write a stored procedure named CountProduct to calculate the number of different products in an order with OrderID (nvarchar(255)) is input parameter and the NbProducts (int) is the output parameter of the procedure.

For example, when we execute the following code, the result should be 1:

declare @t int

exec CountProduct 'CA-2014-100391', @t output

print @t

**Question 9:**

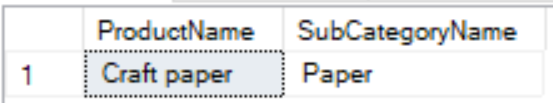
Create a trigger InsertProduct which will be activated by an insert statement into the Product table. The trigger will display the ProductName and the SubCategoryName of the products which have just been inserted by the insert statement.

For example, when we run:

insert into Product(ProductName, UnitPrice, SubCategoryID)

values ('Craft paper', 0.5, 3)

The result should be as follows:



Picture 9.1

**Question 10:**

Insert the following information:

- A category named 'Sports' into table Category

- A subcategory named 'Tennis' and a subcategory named 'Football' into table SubCategory, both these two subcategories are subcategories of Category 'Sports'

-- 2. Write a query to display all customers who are

-- ‘Consumer’ and are from Arlington city as follows:

select \* from Customer c

where c.Segment = 'Consumer' and c.City = 'Arlington'

-- 3. Write a query to display all customers having CustomerName

-- starting with B and placed orders in December 2017.

-- Display the result by descending order of Segment and then by ascending order of CustomerName.

select c.ID, c.CustomerName, c.Segment, c.Country, c.City,

c.State, c.PostalCode, c.Region

from Customer c, Orders o

where c.CustomerName like 'B%'

and c.ID = o.CustomerID and o.OrderDate between '2017/12/1' and '2017/12/31'

order by c.Segment DESC, c.CustomerName ASC

-- 4. Write a query to display SubCategoryID, SubCategoryName and the corresponding

-- number of products (NumberOfProducts) in each sub-category

-- having the number of products greater than 100, by descending order of NumberOfProducts.

select p.SubCategoryID, s.SubCategoryName, count(\*) as NumberOfProducts from Product p, SubCategory s

where s.ID = p.SubCategoryID

group by p.SubCategoryID, s.SubCategoryName

having count(p.SubCategoryID) >100

order by NumberOfProducts DESC

-- 5. Write a query to display ProductID, ProductName, Quantity

-- of all products which have the highest Quantity in one order.

select o.ProductID, p.ProductName, o.Quantity from OrderDetails o, Product p

where o.ProductID = p.ID and o.Quantity = (select max(o.Quantity) from OrderDetails o)

-- Question 6: Write a query to display CustomerID, CustomerName and

-- the number of orders (NumberOfOrders) of customers who have the highest number of orders.

select o.CustomerID, c.CustomerName, count(\*) as NumberOfOrders from Orders o, Customer c

where c.ID = o.CustomerID

group by o.CustomerID, c.CustomerName

having count(o.CustomerID) = (select top(1) count(\*) as s from Orders o, Customer c

where c.ID = o.CustomerID

group by o.CustomerID

order by s DESC)

-- Question 7: Display 5 products with the highest unit prices and

-- 5 products with the smallest unit prices as follows:

select \* from

(select top(5) \* from Product p

order by p.UnitPrice desc) as s

union

select \* from (

select top(5) \* from Product p

order by p.UnitPrice) as s1

-- Question 8: Write a stored procedure named CountProduct to

-- calculate the number of different products in an order with OrderID (nvarchar(255))

-- is input parameter and the NbProducts (int) is the output parameter of the procedure.

drop proc CountProduct

create proc CountProduct @OrderID nvarchar(255), @NbProducts int output

as

begin

declare @count int

select count(\*) from OrderDetails o

where o.OrderID = @OrderID

group by o.OrderID

set @NbProducts = @count

end

declare @t int

exec CountProduct 'CA-2014-100391', @t output

print @t

-- 9. Create a trigger InsertProduct which will be activated by an

-- insert statement into the Product table. The trigger will display

-- the ProductName and the SubCategoryName of the products which have just been

-- inserted by the insert statement.

drop trigger InsertProduct

create trigger InsertProduct

on Product instead of insert

as

begin

select i.ProductName, i.SubCategoryID from inserted i

end

insert into Product(ProductName, UnitPrice, SubCategoryID)

values ('Craft paper', 0.5, 3)

-- 10 ----

insert into Category(CategoryName)

values('Sports')

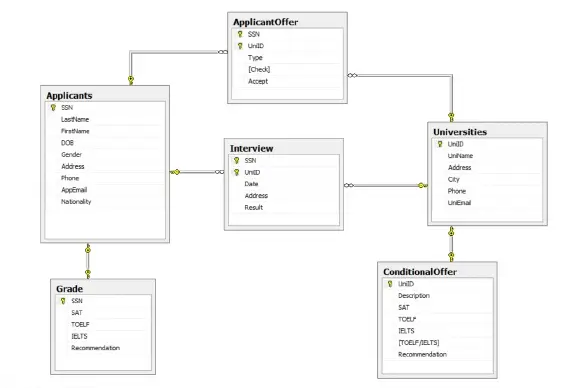
insert into SubCategory values

('Tennis', (select c.ID from Category c where c.CategoryName = 'Sports')),

('Football', (select c.ID from Category c where c.CategoryName = 'Sports'))

**PE 3**

**Question 1:**



Relational Schema

Universities (UniID, UniName, Address, City, Phone, UniEmail)

Applicants (SSN, LastName, FirstName, DOB, Gender, Address, Phone, Email,Nationality)

ApplycantOffer (SSN, UniID)

Interview (SSN, UniID , Date, Address, Result)

ConditionalOffer (UniID, Description, SAT, TOELF, IELTS, TOELF/IELTS,Recommendation)

Grade (SSN, SAT, TOELF, IELTS, Recommendation)

# Set of database statements

CREATE DATABASE [DBI202\_FALL2022\_PROJECT];

CREATE TABLE "Universities" (

"UniID" nvarchar(20) NOT NULL , "UniName" nvarchar (50) NOT NULL , "Address" nvarchar (60) NULL ,

"City" nvarchar(30),

"Phone" nvarchar (24) NULL , "UniEmail" nvarchar(60),

CONSTRAINT "PK\_Universities" PRIMARY KEY CLUSTERED (

"UniID" ASC

)

)

CREATE TABLE "Applicants" (

"SSN" "int" NOT NULL ,

"LastName" nvarchar (50) NOT NULL , "FirstName" nvarchar (30) NOT NULL , "DOB" date NULL ,

"Gender" bit,

"Address" nvarchar (60) NULL , "Phone" nvarchar (24) NULL , "AppEmail" nvarchar(60), "Nationality" nvarchar (30) NULL ,

CONSTRAINT "PK\_Student Details" PRIMARY KEY CLUSTERED (

"SSN" ASC

),

CONSTRAINT "CK\_Birthdate" CHECK (DOB < getdate())

)

CREATE TABLE "ApplicantOffer" (

"SSN" "int" NOT NULL ,

"UniID" nvarchar(20) NOT NULL , "Type" nvarchar(20) NOT NULL, "Check" bit,

"Accept" bit,

CONSTRAINT "PK\_StuUniOffer" PRIMARY KEY CLUSTERED (

"SSN" ASC, "UniID" ASC

),

CONSTRAINT "FK\_AppliOffer\_Stu" FOREIGN KEY (

"SSN"

) REFERENCES "dbo"."Applicants" ( "SSN"

),

CONSTRAINT "FK\_AppliOffer\_Uni" FOREIGN KEY (

"UniID"

) REFERENCES "dbo"."Universities" ( "UniID"

),

CONSTRAINT "CK\_Type\_Apply" CHECK ("Type" IN ('Condition','Uncondition'))

)

CREATE TABLE "Interview" (

"SSN" "int" NOT NULL ,

"UniID" nvarchar(20) NOT NULL , "Date" date,

"Address" nvarchar(50), "Result" nvarchar(50),

CONSTRAINT "PK\_Interview" PRIMARY KEY CLUSTERED (

"SSN" ASC, "UniID" ASC

),

CONSTRAINT "FK\_Interview\_Stu" FOREIGN KEY (

"SSN"

) REFERENCES "dbo"."Applicants" ( "SSN"

),

CONSTRAINT "FK\_Interview\_Uni" FOREIGN KEY (

"UniID"

) REFERENCES "dbo"."Universities" ( "UniID"

)

)

CREATE TABLE "ConditionalOffer" (

"UniID" nvarchar(20) NOT NULL ,

"Description" nvarchar(150), "SAT" float,

"TOELF" float, "IELTS" float, "TOELF/IELTS" int,

"Recommendation" bit,

CONSTRAINT "PK\_ConditionalOffer" PRIMARY KEY CLUSTERED (

"UniID" ASC

),

CONSTRAINT "FK\_ConditionalOffer\_UniNo" FOREIGN KEY (

"UniID"

) REFERENCES "dbo"."Universities" ( "UniID"

),

CONSTRAINT "CK\_TOELF\_IELTS" CHECK ("TOELF/IELTS" IN (1,2)),

)

CREATE TABLE "Grade" (

"SSN" "int" NOT NULL ,

"SAT" float, "TOELF" float, "IELTS" float,

"Recommendation" bit,

CONSTRAINT "PK\_Transcript" PRIMARY KEY CLUSTERED (

"SSN" ASC

),

CONSTRAINT "FK\_Transcript\_Stu" FOREIGN KEY (

"SSN"

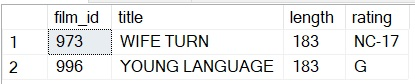
) REFERENCES "dbo"."Applicants" ( "SSN"

)

)

**Question 2:**

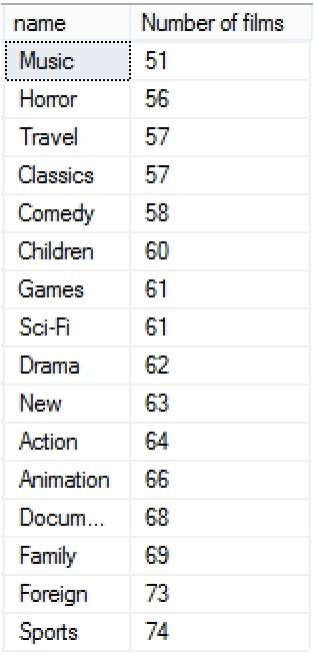
Write a SELECT query to display all films which type is “Documentary” and have length greater than or equals to 180, order by film\_id



Picture 2.1

**Question 3:**

Write a query to display number of films for each category, Order by number of films as ascending.



Picture 3.1

**Question 4:**

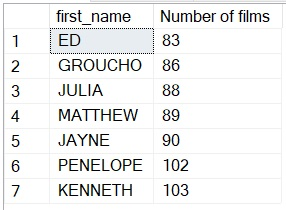
Write a query to display the category which have minimum number of films in it.



Picture 4.1

**Question 5:**

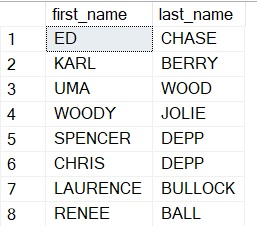
Write a query to display number of films for each actor whose joined more than 80 films, order by number of films as ascending.



Picture 5.1

**Question 6:**

Use subqueries to display all actors who appear in the film Alone Trip.



Picture 6.1

**Question 7:**

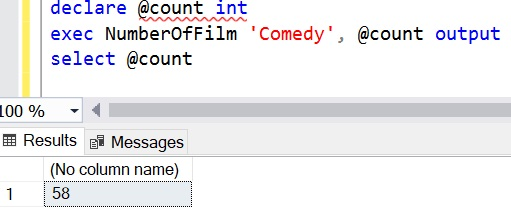
Write a store procedure NumberOfFilm(@catName varchar(25), @count int output), the store procedure is used to count number of films of given category name, the result must be set to @count

If you test your above store as below sample, you can get the those output:

declare @count int

exec NumberOfFilm 'Comedy', @count output

select @count



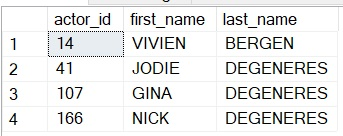
Picture 7.1

**Question 8:**

Write a trigger to make sure that whenever users insert a new film to Film table, the title of film cannot be duplicated.

**Question 9:**

Find all actors whose last name contain the letters GEN:



Picture 9.1

**Question 10:**

You need to find the ID number, first name, and last name of an actor, of whom you know only the first name, "Joe." What is one query would you use to obtain this information.



Picture 10.1

PE 3 – Solution