**SQL SERVER - TASKS**

**TASK - 1**

CREATE TABLE Trainees (

Trainee\_ID INT PRIMARY KEY,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT,

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

Insert 20 sets of records for the above created table based on the requirements in the below queries and perform the following queries.

1. Retrieve all FIRST\_NAME STARTING WITH J-T and should differentiate between Uppercase and lowercase.

2. Retrieve all SALARY BETWEEN 20000 TO 50000

3. Retrieve all FIRST\_NAME ending with i

4. Retrieve all salary without duplications

5. Retrieve all records whose department is Developer and Designer

6. Retrieve all Trainee\_ID less than 5

7. Limit the records by retrieving the 6 to 15 records

8. Retrieve the top 5 records with Ties

9. Retrieve the records in descending order based on department column.

10. Retrieve all last\_name with 3rd character as 'a.'

**TASK – 2**

Perform the following queries:

1. Create a table with primary key, unique key, check and default constraints.

2. Rename a table, database & schema.

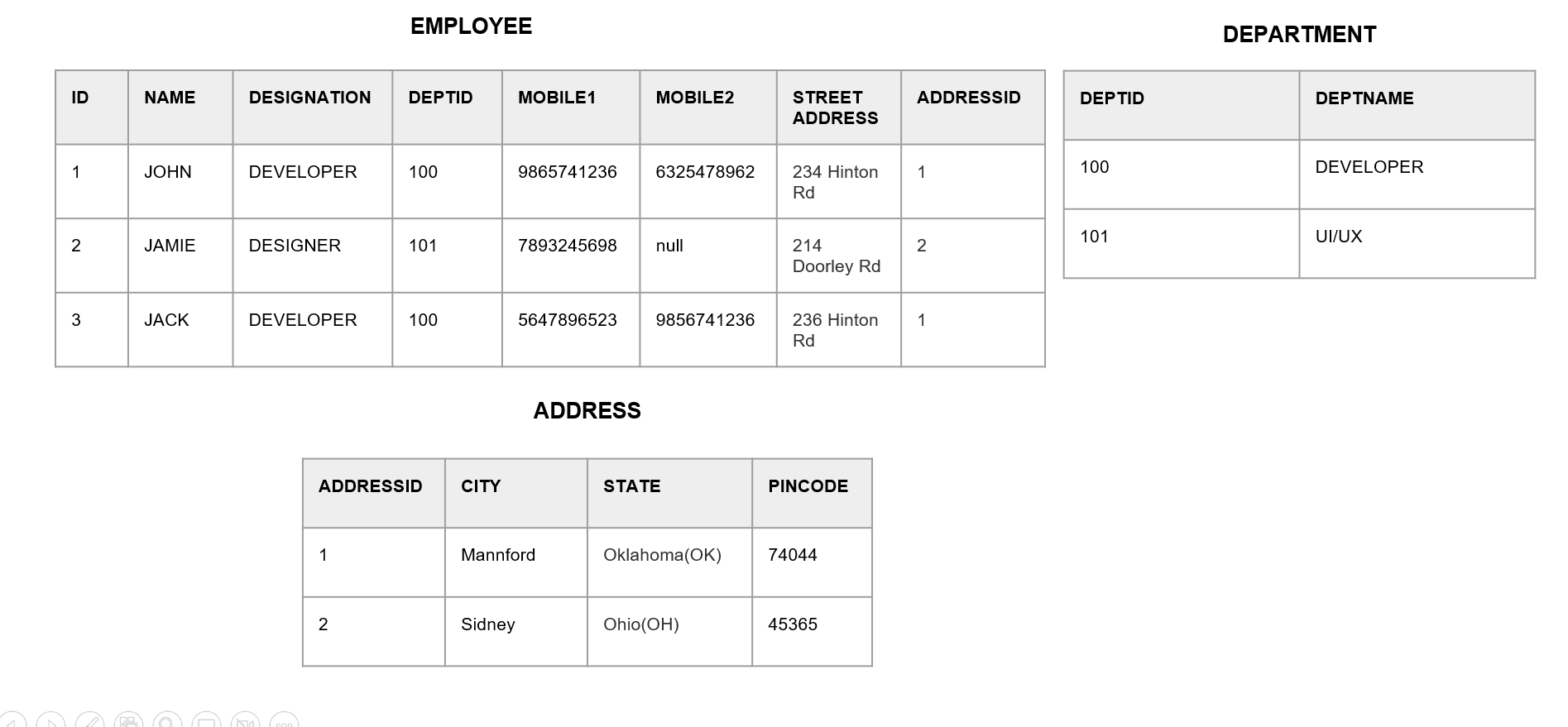
3. Create a table with the following fields:

a. id as identity column, name, salary, increment, computed column as Revised salary(sal+inc)

b. Retrieve all the records whose salary is >20000 and move them into a new table using (SELECT INTO)

**TASK – 3**

Show the working of Primary and Foreign key for the tables in the attached image.

****

**TASK – 4**

* Kindly find the updated attached docx for the task.

USE TRAINEES

CREATE TABLE Worker (

WORKER\_ID INT NOT NULL PRIMARY KEY,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT,

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Worker

(WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY,

JOINING\_DATE, DEPARTMENT) VALUES

(002, 'Niharika', 'Verma', 80000, '2000-02-02', 'Admin'),

(003, 'Vishal', 'Singhal', 300000, '2000-02-02', 'HR'),

(004, 'Amitabh', 'Singh', 500000, '2000-02-02', 'Admin'),

(005, 'Vivek', 'Bhati', 500000, '2000-02-02', 'Admin'),

(006, 'Vipul', 'Diwan', 200000, '2000-02-02', 'Account'),

(007, 'Satish', 'Kumar', 75000, '2000-02-02', 'Account'),

(008, 'Geetika', 'Chauhan', 90000, '2000-02-02', 'Admin');

CREATE TABLE Bonus (

WORKER\_REF\_ID INT, BONUS\_AMOUNT INT, BONUS\_DATE DATETIME, FOREIGN KEY (WORKER\_REF\_ID)

REFERENCES Worker(WORKER\_ID)

ON DELETE CASCADE);

INSERT INTO Bonus

(WORKER\_REF\_ID, BONUS\_AMOUNT, BONUS\_DATE) VALUES

(001, 5000, '2016-02-02'),

(002, 3000, '2016-06-01'),

(003, 4000, '2016-02-02'),

(001, 4500, '2016-02-02'),

(002, 3500, '2016-06-01');

CREATE TABLE Title (

WORKER\_REF\_ID INT, WORKER\_TITLE CHAR(25),

AFFECTED\_FROM DATETIME, FOREIGN KEY (WORKER\_REF\_ID)

REFERENCES Worker(WORKER\_ID) ON DELETE CASCADE

);

INSERT INTO Title

(WORKER\_REF\_ID, WORKER\_TITLE, AFFECTED\_FROM) VALUES

(001, 'Manager', '2016-02-20 00:00:00'),

(002, 'Executive', '2016-06-11 00:00:00'),

(008, 'Executive', '2016-06-11 00:00:00'),

(005, 'Manager', '2016-06-11 00:00:00'),

(004, 'Asst. Manager', '2016-06-11 00:00:00'),

(007, 'Executive', '2016-06-11 00:00:00'),

(006, 'Lead', '2016-06-11 00:00:00'),

(003, 'Lead', '2016-06-11 00:00:00');

Q-1. Write an SQL query to fetch “FIRST\_NAME” from Worker table using the alias name as <WORKER\_NAME>.

Q-2. Write an SQL query to fetch “FIRST\_NAME” from Worker table in upper case.

Q-3. Write an SQL query to fetch unique values of DEPARTMENT from Worker table.

Q-4. Write an SQL query to print the first three characters of FIRST\_NAME from Worker table.

Q-5. Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length.

Q-6. Write an SQL query to print the FIRST\_NAME and LAST\_NAME from Worker table into a single column COMPLETE\_NAME. A space char should separate them.

Q-7. Write an SQL query to print details for Workers with the first name as “Vipul” and “Satish” from Worker table.

Q-8. Write an SQL query to print details of Workers with DEPARTMENT name as “Admin”.

Q-9. Write an SQL query to print details of the Workers whose FIRST\_NAME ends with ‘a’.

Q-10. Write an SQL query to print details of the Workers whose FIRST\_NAME contains ‘a’.

**TASK – 5**

* Kindly find the attached doc for Task 5 in SQL Server.

CREATE TABLE Trainees (

TRAINEE\_ID INT NOT NULL PRIMARY KEY,

FIRST\_NAME CHAR(25),

LAST\_NAME CHAR(25),

SALARY INT,

JOINING\_DATE DATETIME,

DEPARTMENT CHAR(25)

);

INSERT INTO Trainees

(TRAINEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY,

JOINING\_DATE, DEPARTMENT) VALUES

(002, 'Niharika', 'Verma', 80000, '2023-03-20', 'Admin'),

(003, 'Vishal', 'Singhal', 300000, '2023-03-20', 'HR'),

(004, 'Amitabh', 'Singh', 500000, '2023-03-20', 'Admin'),

(005, 'Vivek', 'Bhati', 500000, '2023-03-20', 'Admin'),

(006, 'Vipul', 'Diwan', 200000, '2023-03-20', 'Account'),

(007, 'Satish', 'Kumar', 75000, '2023-03-20', 'Account'),

(008, 'Geetika', 'Chauhan', 90000, '2023-03-20', 'Admin');

1. Write an SQL query to get the count of employees in each department.
2. Write an SQL query to calculate the estimated induction program day for the trainees from 5 days from JOINING\_DATE.
3. Write an SQL query to retrieve the month in words from the Trainees table - JOINING\_DATE Column.
4. Write an SQL query to perform the total and subtotal of salary in each department.
5. Write an SQL query to retrieve first 3 records randomly.
6. Show the working of composite key with any example.
7. Show the working of IIF and CASE for the above table.
8. Show the working of Sequence.
9. Show the working of creation of Synonym for a table in DB1 from DB2.
10. Show the working of IDENTITY\_INSERT.

**TASK - 6**

Kindly find the attached doc for SQL Task 6.

*Sample table*: salesman

salesman\_id | name | city | commission

-------------+------------+----------+------------

5001 | James Hoog | New York | 0.15

5002 | Nail Knite | Paris | 0.13

5005 | Pit Alex | London | 0.11

5006 | Mc Lyon | Paris | 0.14

5007 | Paul Adam | Rome | 0.13

5003 | Lauson Hen | San Jose | 0.12

*Sample table*: customer

customer\_id | cust\_name | city | grade | salesman\_id

-------------+----------------+------------+-------+----------

3002 | Nick Rimando | New York | 100 | 5001

3007 | Brad Davis | New York | 200 | 5001

3005 | Graham Zusi | California | 200 | 5002

3008 | Julian Green | London | 300 | 5002

3004 | Fabian Johnson | Paris | 300 | 5006

3009 | Geoff Cameron | Berlin | 100 | 5003

3003 | Jozy Altidor | Moscow | 200 | 5007

3001 | Brad Guzan | London | | 5005

Sample table: orders

ord\_no purch\_amt ord\_date customer\_id salesman\_id

---------- ---------- ---------- ----------- -----------

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70012 250.45 2012-06-27 3008 5002

70011 75.29 2012-08-17 3003 5007

70013 3045.6 2012-04-25 3002 5001

1. From the above tables write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust\_name and city.
2. From the above tables write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission.
3. From the above tables write a SQL query to find the details of an order. Return ord\_no, ord\_date, purch\_amt, Customer Name, grade, Salesman, commission.
4. From the above tables write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord\_no, purch\_amt, cust\_name, city.
5. From the above tables write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer\_id.

**TASK – 7**

Create a table with 25 records for various departments using the following details:

studentid(pk), studentname, department, score

1. Create a non-clustered index for department.

2. Create a filtered index for department='BCA'

3. Create a view for students in BCA department.

4. Apply Rank() for all the students based on score.

5. Apply Dense\_Rank() for students in each department based on score.

b. Create 2 tables Manager(id(pk), name) and Employee(eid(pk),ename,mid(fk), department).

1. Create a complex view by retrieving the records from Manager and Employee table.

2. Show the working of 'on delete cascade on update set default' for the above tables

**TASK – 8**

Create User-define Scalar function to calculate percentage of all students after creating the following table.

(Create a table with studentid, studentname, semester, securedmarks, totalmarks)

Sample: 1,'John','Sem 1',450,500

2. Create user-defined function to generate a table that contains result of Sem 3 students.

3. Write SQL stored procedure to retrieve all students details. (No parameters)

4. Write SQL store procedure to display Sem 1 students details. (One parameter)

5. Write SQL Stored Procedure to retrieve total number of students details. (OUTPUT Parameter)

6. Show the working of Merge Statement by creating a backup for the students details of only students in Sem 1.

Note: Update few values in students details so that you can see the working of UPDATE.

**TASK – 9**

1. Write a Stored Procedure in SQL using conditional statements to search for a record from the students table (created in SQL Task 8) based on studentname column.

2. Write a Stored procedure in SQL to give remarks for the secured marks column in the students table (created in SQL Task 8) using CASE statement.

3. Show the working of Table variables, temporary table, temporary stored procedures. (Both Local and Global)

**TASK – 10**

Create a table for employees (EMPLOYEE\_ID | FIRST\_NAME | LAST\_NAME | EMAIL | PHONE\_NUMBER | HIRE\_DATE | JOB\_ID | SALARY) with 20 records (Include 20 trainees in our training session)

1. Write a SQL query to find those employees who receive a higher salary than the employee with ID 16. Return first name, last name.

2. Write a SQL query to find out which employees have the same HIRE\_DATE as the employee whose ID is 11. Return first name, last name, job ID.

3. Write a SQL query to find those employees who earn more than the average salary. Return employee ID, first name, last name.

4. Write a SQL query to find those employees who report to that manager whose first name is ‘’Yamini". Return first name, last name, employee ID and salary.

5. Write a SQL query to find those employees whose salary falls within the range of the smallest salary and 2500. Return all the fields.

**TASK – 11**

1. Show the working of PIVOT and UNPIVOT by creating a table students(sid,sname,department,yearofcompletion) with 20 records.

Note: Simplify the table to show the number of students in each department and categorize them based on yearofcompletion.

2. Create a database for Library Management. Analyse the required columns, perform a normalization and link the tables using primary and foreign key (using on update set default)

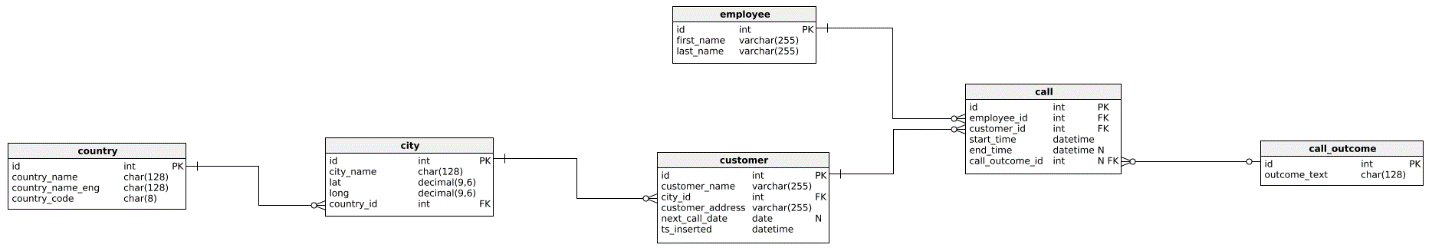
Note: Kindly do the task on your own.

**TASK – 12**

1. Kindly refer the attached image and create the tables and link them accordingly.

2. Create a Database diagram for the created tables.

Note: Create a separate database for the tables.



**TASK – 13**

a. Create a table Hobbies (HobbyID(pk), HobbyName(uk))

1. insert records into the table using a stored procedure.

2. insert duplicate records into the table and show the working of exception handling using Try/catch blocks.

3. Store the error details in an errorbackup table.

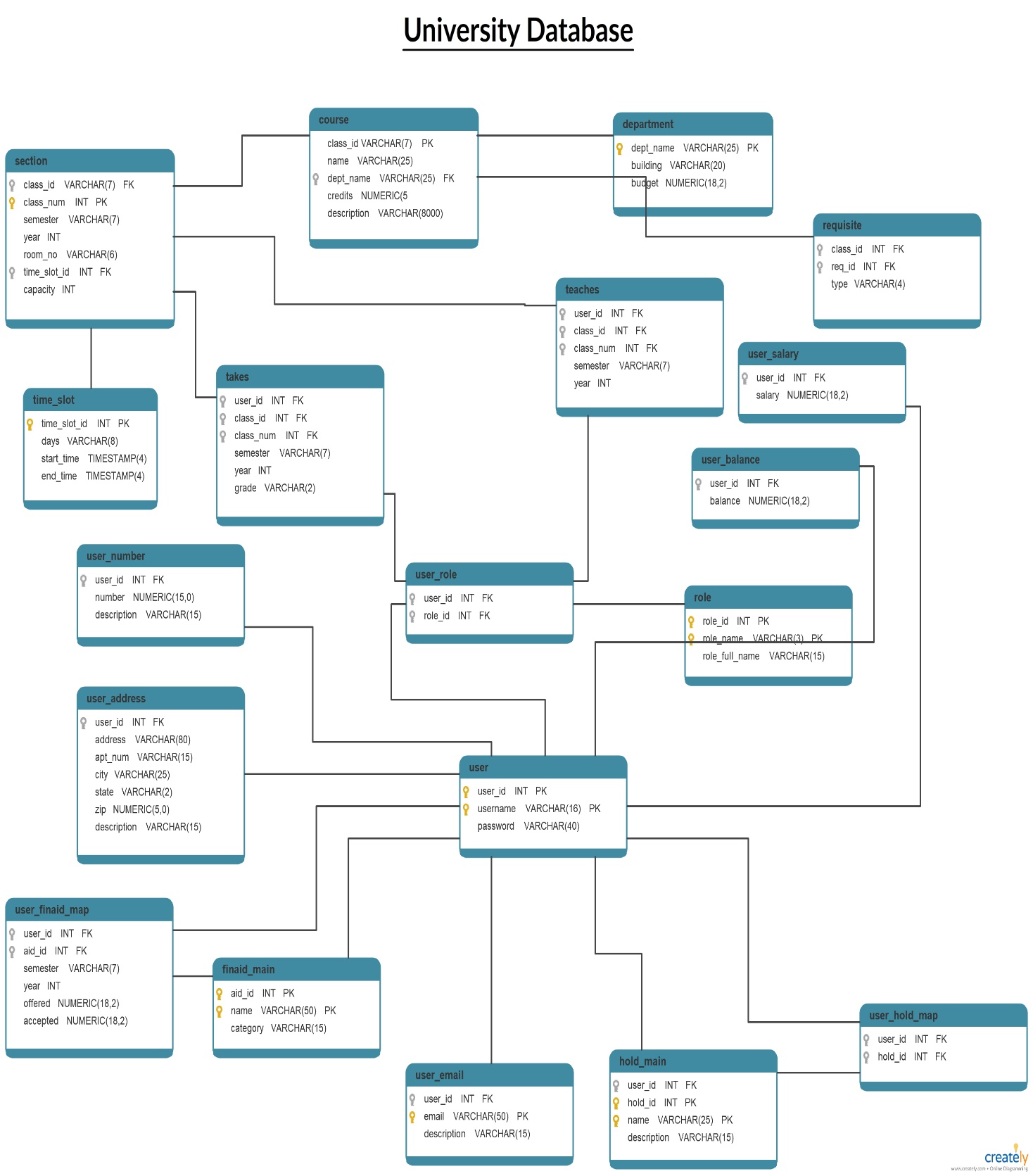
b. Create a procedure to accept 2 numbers, if the numbers are equal then calculate the product else use RAISERROR to show the working of exception handling.

c. Create a table Friends(id(identity), name (uk)) and insert records into the table using a stored procedure.

Note: insert the names which start only with A,D,H,K,P,R,S,T,V,Y ELSE using THROW give the error details.

**TASK – 14**

1. Kindly refer the attached image and create the tables and link them accordingly.  
   2. Create a Database diagram for the created tables.  
     
   Note: Create a separate database for the tables.



**TASK – 15**

Show the working of the following concepts for the students table created in the earlier tasks.

1. Auto-commit and Auto-rollback transactions.

2. Implicit transactions

3. Explicit transactions

a. Only Commit - insert statement

b. Only Rollback - update statement

c. Savepoint - commit update and insert statements, rollback delete statement

**TASK – 16**

1. Create a DML trigger to restrict DML operations on Saturday and Sunday.

2. Create a DML trigger to restrict delete operations between 11:00AM to 15:00PM.

3. Create a DDL trigger to show notification whenever a CREATE, ALTER, DROP, RENAME operation is performed.

**TASK – 17**

Kindly refer the attached .txt file and perform the following:

1. Analyze every database and list down how many tables, what tables with columns, and how each table is related to other tables.

2. Kindly make used of modified\_date, referential integrity (on update set default), soft delete wherever necessary.

**Note:** No need of performing any queries, only analysis is enough.

College

School

Hospital

Pet Clinic

Children's Clinic

Mall

Training Institution

E-commerce website

E-banking website

Sports Academy

Theme Park

E-learning website

Restaurant

IT Company

Movie ticket website

Travel agency website

Gym management

Car Rental management

Job Portal

Bus Booking management

Real-estate management

Crime-reporting management