Project On

ShowManHouse

Developed by

Name : *Zahra Arshadi*

Reg.No: *(R113021600046)*
Faculty: *Mr. Sahami*

NIIT

Tehran West Center
30216

ShowmanHouse

(Project Title)

Batch Code: *B110019*

Name of the Coordinator: *Mr. Sahami*

Name of Developers: *Zahra Arsadi , Hilda Fazel*

NIIT

CERTIFICATE

This is the certify that this report, titled Showman House ,embodies the original work done by Zahra Arshadi and Hilda Fazel in partial fulfillment of his course requirement at NIIT.

Coordinator:

Mr. Sahami

ACKNOWLEDGMENT

We have benefited a lot from the feedback and suggestion given to us by Mr. Sahami and other faculty members.

System Analysis

System summary

Showman House is a very large event management company in South America. It has various offices in Peru, Chile, and Argentina and the head office is in Brazil. Showman House is well-known for its efficiency, good-quality staff, and affordable charges. The company manages various types of events throughout the year. These events include fashion shows, celebrity shows, chat shows, musical extravaganzas, exhibitions, fairs, and charity shows.

Because the bussiness is growing, the Showman House is unable to maintain the details manually. At any given point of time, Showman House manages at least 20 events. Maintaining all the information about these events manually is difficult and time-consuming

To create the computerized event management system for Showman House, the project team of CreateMyDb Inc. needs to perform the following tasks:

- Create a database called ShowmanHouse.
- Create the table designs as per the relationship diagram ensuring minimum disk space utilization.
- Perform validation on the tables as per the requirements.
- Create appropriate relationships between the tables. Store the details of all the employees who have managed an event in the current month in a text file.

ACKNOWLEDGMENT

This information will be displayed on the organization's website. Make use of the required tools to perform the data transfer.

- Create appropriate indexes to speed up the execution of the following tasks:
- Extracting the customer details for an event organized on a particular date.
- Extracting event details for all the events where the payment is pending.
- Displaying the details of all the events where the staff required is greater than 25.
- Implement a proper security policy in the database. For this, create logins named William, Sam, Chris, and Sara. Chris is the database administrator and William, Sam, and Sara are database developers.
- Back up the database daily and store the backup in the C drive.
- Store crucial data in encrypted format.
- Ensure that an alert is sent to Chris whenever the size of the temporary space in the database falls below 20 MB.

Data Base Design

Database Name: Students.mdb

Number of Schemas: 4

Schema Names:

1. HumanResources
2. Managements
3. Events
4. Information

Number of Tables: 3

Table Names:

- ✓ HumanResources.Employee
- ✓ Managements.Events
- ✓ Managements.Payment
- ✓ Managements.PaymentMethod
- ✓ Management.credit
- ✓ Management.check
- ✓ Events.Customer
- ✓ Events.EventType
- ✓ Information.City
- ✓ Information.Phone

Shematic Diagram Of The Database

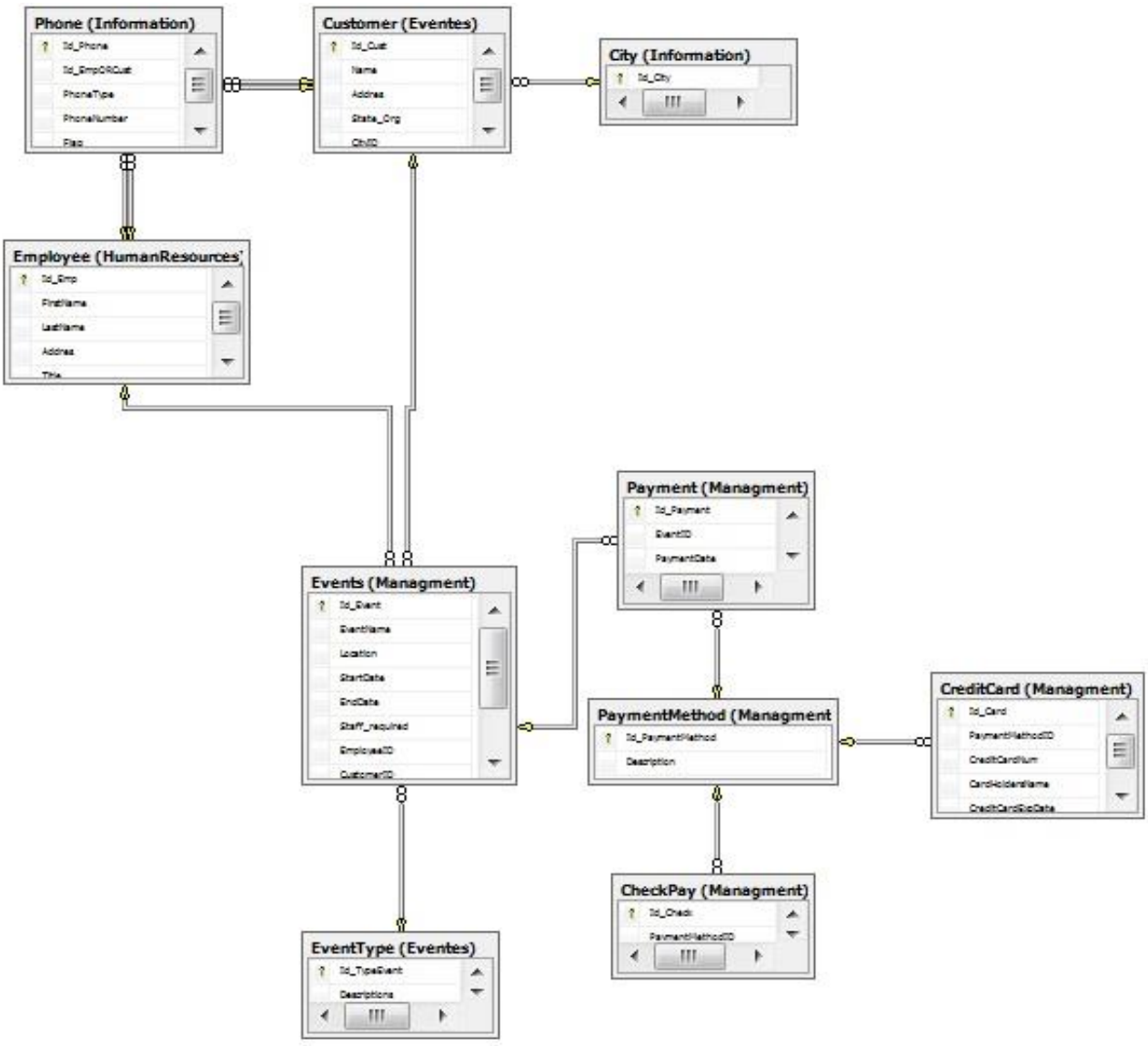

Table Design

Table: Management.Events

Field Name	Data type	Size	Description
Id_Event	Int		Event number
EventName	Nvarchar	50	Event name
Location	nvarchar	50	Place for organize Event
StartDate	Date		Start date of the Event
EndDate	Date		End date of the Event
Staff_Required	int		Number of Staff required
EmployeeID	int		Number of Employee
CustomerID	int		Number of Customer
EventTypeID	int		Event type of Event
NumOfPeople	int		Number of gueste

Table:Management.EventType

Field Name	Data type	Size	Description
Id_TypeEvent	int		Number of type Event
Description	Nvarchar	50	Class of the student
CharegePerPerson	int		Charege per person

Table Design

Table:HumanResources.Employee

Field Name	Data type	Size	Description
Id_Emp	int		Number of Employee
FirstName	navrchar	50	Name of Employee
LasteName	nvarchar	50	LastName of Employee
Addres	nvarchar	50	Address of Employee
Title	nvarchar	50	Title of Employee

Table : Eventes.Customer

Field Name	Data type	Size	Description
Id_Cust	int		Number of customer
Name	Nvarchar	50	Name of customr
Addres	Nuvarchar	50	Addres of customer
State_Org	Nvarchar	50	State
CityID	int		Number of customer's city

Table Design

Table: Management.PaymentMethod

Field Name	Data type	Size	Description
Id_PaymentMethod	int		number of the PaymentMethod
Description	Nvarchar	50	Description of type payment

Table : Management.Payment

Field Name	Data type	Size	Description
Id_Payment	int		Number of Payment
EventID	int		Number of Event
PaymentDate	Date		Date od payment
PaymentMethodID	int		Number of payment method
PaymentAmount	int		Amount of Payment

Table: Management.CheckPay

Field Name	Data type	Size	Description
Id_Check	int		number of check
PaymentMethodID	int		Number of payment method
CheckNumber	int		Number of check

Table Design

Table:Management.CreditCard

Field Name	Data type	Size	Description
Id_Card	int		Number of card
PaymentMethodID	int		Number of payment method
CreditCardNumber	nvarchar	50	Number of credit card
CardHoldersName	nvarchar	50	Name of card holder
CreditCardExpDate	nvarchar	50	Expire date of credit card

Table: Information.Phone

Field Name	Data type	Size	Description
Id_Phone	int		Number of customer
Id_EmpORCust	int		Name of customr
PhoneType	Nuvarchar	50	Addres of customer
PhoneNumber	Nvarchar	50	State
Flag	bit		Number of customer's city

Table:Information.City

Field Name	Data type	Size	Description
Id_City	int		number of City
NameCity	Nvarchar	20	Namer of city

Validations Performed

Table: Management.Events

Validation Required	Methods Used For Validation
Id_Event must be unique	Primary key constraint
Id_Event must be generated	Identity specification
EventName ,Location , StartDate , EndDate, Staff_Required, NumOfPeople should not be left blank.	Use Not null in the create table and ltrim function in create procedure
Staff_Requred should be greater than 0	CHECK Constraint
StartDate shuld be less than the EndDate	CHECK Constraint
StartDate and EndDate should be greater than current date	CHECK Constraint
NumOfPeople should be greater than or equal to 50	CHECK Constraint
EventTypeID,CustomerID,EmployeeID is a foreign key	Foreign Key Constraint

Table:Management.EventType

Validation Required	Methods Used For Validation
Id_TypeEvent must be auto generated	Identity key Constraint
Description should not be left blank	Use Not null in the create table and ltrim function in create procedure
CharegePerPerson should be greater than 0	CHECK Constraint

Validations Performed

Table:HumanResources.Employee

Validation Required	Methods Used For Validation
Id_Emp should be auto generated	Identity specification
FirstName ,LastName , Address , should not be left blank.	Use Not null in the create table and ltrim function in create procedure
Title should have one of values: Executive, Senior Executive, Management Trainee, Event Management, Senior Event Management	CHECK Constraint

Table : Eventes.Customer

Validation Required	Methods Used For Validation
Id_Cust should be auto generated	Identity specification
Name , Address ,City ,State should not be left blank.	Use Not null in the create table and ltrim function in create procedure

Table: Management.PaymentMethod

Validation Required	Methods Used For Validation
Id_PaymentMethod should be auto generated	Identity specification
Description must contain any of three values: cash, cheque ,credit	CHECK Constraint

Validations Performed

Table : Management.Payment

Validation Required	Methods Used For Validation
Id_Payment must be auto generated	Identity key Constraint
PaymentDate should be less than or equal to start date of the event	Trigger
PaymentDate cannot be less than the current date	Trigger
PaymentMethodID is foreign key from PaymentMethod table	CHECK Constraint
PaymentAmount=ChaargePerPerson * NumOfPeople	procedure

Table: Management.CheckPay

Validation Required	Methods Used For Validation
Id_Check must be auto generated	number of check
PaymentMethodID is foreign key from PaymentMethod table	Foreign Key Constraint
CheckNumbe should be left blank	Use Ltrim function

Table Design

Table:Management.CreditCard

Validation Required	Methods Used For Validation
Id_Card must be auto generated.	Identity key Constraint
PaymentMethodID is foreign key from PaymentMethod table	Foreign Key Constraint
CreditCardExpDate should be greater than current date	CHECK Constraint

Table: Information.Phone

Validation Required	Methods Used For Validation
Id_Phone must be auto generated	Identity key Constraint
Id_EmpORCust is foreign key from customer or employee tables	Foreign Key Constraint
PhoneType should be values : Mobile, Home, Fax, Work	CHECK Constraint
PhoneNumbershould be entered format [0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]	CHECK Constraint

Table:Information.City

Validation Required	Methods Used For Validation
Id_City must be auto generated	number of City
NameCity should be left blank	Use Ltrim function

Configuration

Hardware: PC compatible with an Intel(R) Core(TM)2 Duo CPU2.53 GHz , 320 GB of hard disk , 4 GB of RAM, DVD/CD_ ROM Drivers

Operating System: Microsoft Seven(64 bit)

Software: Microsoft SQL Server 2007 Standard Edition

PROJECT FILE DETAILS		
S.NO	File Name	Remarks
	ShowmanHouse.mdf ShowmanHouse.ldf	SQL Server database that constraint tables,procedures, triggers, constraints, and queries

Syntax Project

```
--create database ShowmanHouse
--on
--(
--FileName ='D:\NIIT\Q2\project\ShowmanHouse.mdf',
--Name = 'ShowmanHouse.mdf'
--)
--log on
--(
--FileName ='D:\NIIT\Q2\project\ShowmanHouse.ldf',
--Name = 'ShowmanHouse.ldf'
--)
----------------------------Create schema--------------------------------

--create schema HumanResources
--Go
--create schema Managment
--Go
--create schema Eventes
--Go
--Create schema Information
--GO

-------------------------------Create Table Phone-------------------------
--create table Information.Phone
--(
--Id_Phone int identity,
--Id_EmpORCust int not null,
--PhoneType nvarchar(30)not null,
--PhoneNumber nvarchar(max)not null,
--Flag bit,
--constraint PK_Phone primary key (Id_Phone),
--constraint CH_PhonNum check (PhoneNumber like '[0-9][0-9]-[0-9][0-9][0-
9]-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]'),
--constraint CH_Type check (PhoneType in ('Mobile','Home','Fax','Work'))
--)
-------"Insert Phone"-----
--alter proc USP_InsertPhone
--@EmpORCustID int,
--@PhoneType nvarchar(50),
--@PhoneNumber nvarchar(50),
--@Flag bit
--as
--Begin try
--insert into Information.Phone
values(@EmpORCustID,ltrim(@PhoneType),ltrim(@PhoneNumber),@Flag)
--End try
--Begin catch
--Print Error_message()
--end Catch
--Go
```

Syntax Project

```
--------------------------------create table city-------------------------

--create table Information.City
--(
--Id_City int identity,
--NameCity nvarchar(20)not null,
--constraint PK_City primary key(Id_City)
--)
--Go
-------"Insert City"-----

--alter proc USP_InsertCity
--@NameCity Nvarchar(50)
--as
--begin try
--insert into Information.City values(ltrim(@NameCity))
--end try
--begin catch
--print Error_message()
--end catch
--Go--end Catch
--Go
------------------------------Create Table Employee-------------------------
--create table HumanResources.Employee
--(
--Id_Emp int identity,
--FirstName Nvarchar(50)not null,
--LastName nvarchar(50)not null,
--Addres nvarchar(50)not null,
--Title nvarchar(50),
--constraint PK_Emp primary key(Id_Emp),
--constraint ch_title check (title in('Executive','Senior
Executive','Management Trainee','Event Management','Senior Event
Management'))
--)
--Go
-------"Insert Employee"-----
--create proc USP_InsertEmployee
--@firstname nvarchar(50),
--@lasttname nvarchar(50),
--@address nvarchar(50),
--@title nvarchar(50)
--as
--begin try
--insert into HumanResources.Employee
values(ltrim(@firstname),ltrim(@lasttname),ltrim(@address),@title)
--end try
--begin catch
--print Error_message()
--end catch
```

Syntax Project

```
----------------------------Create Table Customer--------------------------------
--create table Eventes.Customer
--(
--Id_Cust int identity,
--Name nvarchar(50)not null,
--Addres nvarchar(50)not null,
--State_Org nvarchar(50)not null,
--CityID int not null,
--constraint PK_Cust primary key(Id_Cust),
--constraint FK_City foreign key(CityID) references Information.City on delete
cascade
--)
--Go
-------"Insert Customer"-----
--create proc USP_InsertCustomer
--@name nvarchar(50),
--@address nvarchar(50),
--@State nvarchar(50),
--@City int
--as
--begin try
--insert into Eventes.Customer
values(ltrim(@name),ltrim(@address),ltrim(@State),ltrim(@City))
--end try
--begin catch
--print Error_message()
--end catch
---------------------------------create Table EventTypeFile-----------------------
--create table Eventes.EventType
--(
--Id_TypeEvent int identity,
--Descriptions nvarchar(100),
--ChargePerPerson int not null,
--constraint PK_TypeEvent primary key(Id_TypeEvent),
--constraint check_Price check(ChargePerPerson>0),
--constraint ch_Type check (Descriptions in('Celebrity show','Fashion show','Chat
show','Musical Extravaganazas','Exhibitions','Fairs','Charity show'))
--)
--Go
-------"Insert EventType"-----

--alter proc USP_InsertEventType
--@des nvarchar(50),
--@charge int
--as
--begin try
--insert into Eventes.EventType values(ltrim(@des),@charge)
--end try
--begin catch
--print Error_message()
--end catch
```

Syntax Project

```
--create proc USP_InsertEventType
--@des nvarchar(50),
--@charge int
--as
--begin try
--insert into Eventes.EventType values(ltrim(@des),@charge)
--end try
--begin catch
--print Error_message()
--end catch
--Go
-----------------------create table Evenet-----------------------------
Create table Management.Events
--(
--Id_Event int identity,
--EventName nvarchar(50)not null,
--Location nvarchar(50) not null,
--StartDate date not null,
--EndDate date not null,
--Staff_required int not null,
--EmployeeID int,
--CustomerID int,
--EventTypeID int,
--NumOfPepole int ,
--constraint PK_Event primary key(Id_Event),
--constraint Check_staff check(Staff_required>0),
--constraint Check_StartDate check(StartDate<EndDate),
--constraint Check_Date1 check(StartDate>=getdate()),
--constraint Check_Date2 check(EndDate>=getdate()),
--constraint FK_Emp foreign key(employeeID) references HumanResources.Employee on
delete cascade,
--constraint FK_Cust foreign key(CustomerID) references Eventes.Customer on delete
cascade,
--constraint FK_TypeEvent foreign key(EventTypeID) references Eventes.EventType on
delete cascade
--)
--Go
--dbcc checkident('Managment.Events',reseed,0)
--Go
```

Syntax Project

```
-------"Insert Events"-----
--create proc USP_InsertEvent
--@name nvarchar(50),
--@Loc nvarchar(50),
--@StartDate Date,
--@EndDate Date,
--@Staff int,
--@EmpID int,
--@CustID int,
--@Eventtype int,
--@number int
--as
--begin try
--insert into Managment.Events
values(ltrim(@name),ltrim(@Loc),ltrim(@StartDate),ltrim(@EndDate),ltrim(@Staff),@Em
pID,@CustID,@Eventtype,ltrim(@number))
--end try
--begin catch
--print Error_message()
--end catch

-------------------------------create PaymentMethod-------------------------
--Create table Managment.PaymentMethod
--(
--Id_PaymentMethod int identity,
--Description nvarchar(20),
--constraint PK_Method primary key(Id_PaymentMethod)
--constraint Ch_Des check (Description in('Check','Cash','Credit Card'))
--)
--Go
-------"Insert PaymentMethod"-----
--create proc USP_PaymentMethod
--@des nvarchar(50)
--as
--begin try
--insert into Managment.PaymentMethod values(ltrim(@des))
--end try
--begin catch
--print Error_message()
--end catch

-------------------------------create Table Payment-------------------------
--create table Managment.Payment
--(
--Id_Payment int identity,
--EventID int ,
--PaymentDate date,
--PaymentMethodID int,
--PaymentAmount int,
--constraint PK_Payment primary key(Id_Payment),
--constraint FK_Method foreign key(PaymentMethodID) references
Managment.PaymentMethod on delete cascade,
--constraint FK_Event foreign key(EventID) references Managment.Events on delete
cascade
--)
```

Syntax Project

```
-------"create Trigger for Paymente"-----
--Create trigger Check_PayDate
--on Managment.Payment
--for insert,update
--as
--declare @PayDate date
--declare @StartEvent date
--select @PayDate=PaymentDate from Managment.Payment
--select @StartEvent=StartDate from Managment.Events
--begin
--       if(@PayDate>=@StartEvent)
--               begin
--               commit tran
--               end
--            else
--                   if (@PayDate>=GETDATE())
--                   begin
--                   commit tran
--                   end
--               else
--                  begin
--                  print 'you can insert becuse youre payment date is loss start
date'
--                  rollback transaction
--                  end
--end
-------"Insert Payment"-----
--create proc USP_InsertPayment
--@EventID int,
--@PayDate date,
--@PayMethod int,
--@Charge int,
--@Numpeople int
--as
--begin try
--select @Charge=ChargePerPerson from Eventes.EventType
--select @Numpeople=NumOfPepole from Managment.Events
--insert into Eventes.Customer
values(@EventID,@PayDate,@PayMethod,@charge*@Numpeople)
--end try
--begin catch
--print Error_message()
--end catch
-------------------------------create Table Credit---------------------

--create table Managment.CreditCard
--(
--Id_Card int identity,
--PaymentMethodID int,
--CreditCardNum nvarchar(50)not null,
--CardHoldersName nvarchar(50) not null,
--CreditCardExpDate nvarchar(50) not null,
--constraint PK_Card primary key(Id_card),
--constraint FK_PayMethod foreign key(PaymentMethodID) references
Managment.PaymentMethod  on delete cascade,
--constraint Check_Date check(CreditCardExpDate>=getdate())
--)
```

Syntax Project

```
-------"Insert Credit"-----
--create proc USP_InsertCredit
--@PayMethod int,
--@CardNum nvarchar(50),
--@CardHolder nvarchar(50),
--@CardExp nvarchar(50)
--as
--begin try
--open symmetric key datakey
--decryption by certificate data
--insert into Managment.CreditCard
values(@PayMethod,encryptbykey(key_guid('datakey'),@CardNum),encryptbykey(key_guid(
'datakey'),@CardHolder),encryptbykey(key_guid('datakey'),@CardExp))
--end try
--begin catch
--print Error_message()
--end catch
--------------------------------create Table Check---------------------------------
--create table Managment.CheckPay
--(
--Id_Check int identity,
--PaymentMethodID int,
--CheckNumber int not null,
--constraint PK_Check primary key(Id_Check),
--constraint FK_PayMethod2 foreign key(PaymentMethodID) references
Managment.PaymentMethod  on delete cascade,
--)
-------"Insert Check"-----
--create proc USP_InsertCheck
--@PayMethod int,
--@CheckNum int
--as
--begin try
--open symmetric key datakey
--decryption by certificate data
--insert into Managment.CheckPay
values(@PayMethod,encryptbykey(key_guid('datakey'),ltrim(@CheckNum)))
--end try
--begin catch
--print Error_message()
--end catch

-------------create information employee-------

--bcp select * from HumanResources.Employee E,Managment.Events M where
E.Id_Emp=M.EmployeeID
--queryout d:\Employee.txt -T -t , -c
```

Syntax Project

```
----------------------------create index customer------------------------------
--create unique index Index_Cust
--on Eventes.Customer(Id_Cust)
-------"proc for Index_Cust"-------
--create proc USP_CustForEvent
--@date date
--as
--begin try
--select * from Eventes.Customer c join Managment.Events e on
c.Id_Cust=e.CustomerID
--where @date=e.StartDate
--end try
--begin catch
--print error_message()
--end catch
---------------------------create index Event and Pay-----------------------------
--create unique index Index_event
--on Managment.Events(Id_Event)
--Go
--create unique index Index_Pay
--on Managment.Payment(Id_Payment)
--Go
-------"View for Index_Event and Pay"-----
--create view PayPending
--as
--select * from Managment.Events e join Managment.Payment p on e.Id_Event=p.EventID
--where p.PaymentAmount=0
---------------------------create index Staff---------------------------------
--create nonclustered index Index_Staff
--on Managment.Events(Staff_Required)
-------"View for View"-----
--create view [Staff>25]
--as
--select * from Managment.Events where Staff_required>25
--------------create login name--------------
--create login William
--with password='6666'
--Go
--create user William for login William
--Go
--create login Sam
--with password='1234'
--Go
--create user Sam for login Sam
--Go
--create login Chris
--with password='6565'
--Go
--create user Chris for login Chris
--Go
--create login Sara
--with password='5678'
--Go
--create user Sara for login Sara
```

Syntax Project

```
-------"deny for developer"-----
--deny alter to William
--Go
--deny alter to Sam
--Go
--deny alter to Sara

------------create Backup----------------
--exec sp_addumpdevice 'disk','ShowmanHouse','c:\Backup\ShowmanHouse.bak';
--backup database ShowmanHouse to ShowmanHouse

-----------create encrypte format----------
--create master key
--encryption by password='zhaf'
--Go
--create certificate data
--with subject='Encrypt My Informations'
--Go
--create symmetric key datakey
--with algorithm=AES_256
--encryption by certificate data
```