Procedures:

1. Procedure to find Vacant Units based on Input Date

Create procedure VacantUnits @a Datetime

As

Begin

Select u.UnitCode,ua.UnitID,ua.EndDate,A.maxdt,A.UnitID

From Unit u

Inner Join UnitAllocation ua

on u.UnitID=ua.UnitID

Inner Join

(

Select UnitID,max(EndDate) maxdt

from UnitAllocation

Group by UnitID

)A

on ua.UnitID=A.UnitID

where

A.maxdt<=@a

End

Exec VacantUnits '2019-11-18'

1. Procedure to calculate Rent Increase by default 10% increase every year

Create Procedure CalculateRentIncrease @u int,@dt DateTime,

@rent int Output

As

Begin

Select @rent =

Uprice\*(power(1.100,DATEDIFF(Year,DateCreated,@dt)))

from Unit

where unitID=@u

END

Declare @rent int

Exec CalculateRentIncrease 110 , '2020-01-01', @rent Output

Print @rent

1. Procedure To calculate Rank of Units:

Create procedure CalculateRank @unitID int,

@Rank int OUTPUT

AS

BEGIN

declare @rankCal int

select @Rank =

(select case

when sum(DistanceID) = 5 then 1

when sum(DistanceID) = 6 then 2

when sum(DistanceID) = 7 then 3

when sum(DistanceID) = 8 then 4

else 5

end as CRank

from UnitScore us group by UnitID having us.UnitID = @unitID )

return @Rank

END

1. Procedure to get vendor Information

Create Procedure VendorInformation @vendorID int as

BEGIN

select v.VendorID, v.VendorName, w.UnitID as WorkingOnUnit, w.ContractStartDate,

w.ContractEndDate

from Vendor v join WorksOn w

on v.VendorID = w.VendorID

where v.VendorID = @vendorID

END

Derived Column Function:

1. Function to calculate CreditEligibility Derived Column On Customer Table

Create Function dbo.CheckCreditEligibilityCrieteria(@a int)

returns char(3)

AS

Begin

Declare @check char(3)

Select @check =case when @a>700

Then 'YES'

Else 'NO'

End

return @check

End

Non Clustered Indexes:

CREATE INDEX PropertyID\_IND

ON Property(PropName,PropMgrID)

Create Index enddt\_IND

ON UnitAllocation(EndDate,UnitID,Tenant\_ID)

Create Index Tenant\_Unit\_IND

on Payment(Tenant\_ID,UnitID)

Create Index UnitPropId\_IND

on Unit(PropID)

Create Index TenantDeposit\_IND

on Tenant(TenantDeposit)

Create Index MaintenanceCostVendor\_IND

on MaintenanceCost(VendorID,MaintenanceID)

Create Index UnitVendor\_workson\_IND

on WorksOn(UnitID,VendorID)

Trigger:

1. Trigger on Payment Table

Create Trigger PaymentRecordTracking on Payment

For Update

As

Begin

Insert into PaymentRecords

(

Tenant\_ID ,

TenantName ,

Unit ,

UnitCode,

PaymentDate ,

PaymentAmount ,

ActionDate )

Select p.Tenant\_ID,t.TenantName,p.UnitID,u.UnitCode,p.PaymentDate,p.PaymentAmount,Getdate()

From deleted p

Inner Join Tenant t

on p.Tenant\_ID=t.Tenant\_ID

Inner Join Unit u

on u.UnitID=p.UnitID

End

Column Data Encryption:

1. Encryption on Tenant table

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Password123';

CREATE CERTIFICATE Certificate1

WITH SUBJECT = 'Protect Data';

CREATE SYMMETRIC KEY SymmetricKey1

WITH ALGORITHM = AES\_128

ENCRYPTION BY CERTIFICATE Certificate1;

OPEN SYMMETRIC KEY SymmetricKey1

DECRYPTION BY CERTIFICATE Certificate1;

GO

UPDATE Tenant

SET

TenantCreditCardNumber='876-779-7000',

TenantCreditCardNumber\_encrypt = EncryptByKey (Key\_GUID('SymmetricKey1'),'876-779-7000')

FROM Tenant

Where Tenant\_ID=104

GO

-- Closes the symmetric key

CLOSE SYMMETRIC KEY SymmetricKey1;

Views:

1. View on Tenant Information

CREATE view TenantTerm as

select t.TenantName, t.TenantContact, t.TenantDeposit, u.UnitCode,ua.StartDate, ua.EndDate from tenant as t

join payment as p on t.Tenant\_ID=p.Tenant\_ID

join unitallocation as ua on t.Tenant\_ID=ua.Tenant\_ID

join unit as u on ua.UnitID=u.UnitID

join property as po on u.PropID=po.PropID

Group by tenantname,tenantcontact,TenantDeposit,u.UnitCode,startdate,EndDate;

1. View for Unit Rank

create view UnitRank as

select distinct(UnitID), dbo.getRank(UnitID) as Rank

from UnitScore group by UnitID;