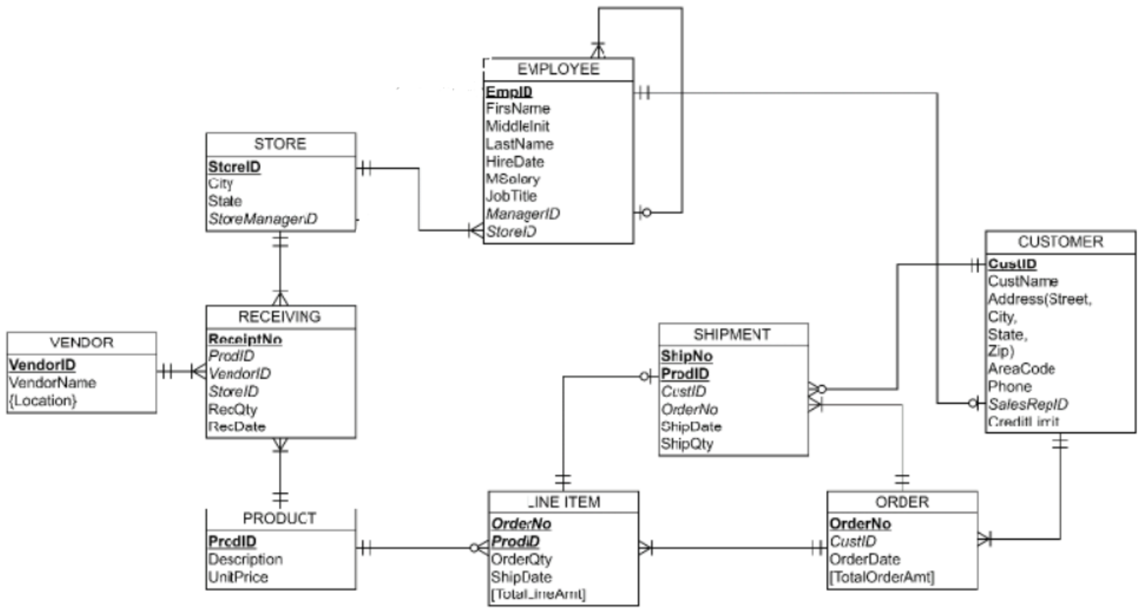
Contents

Crows Feet ERD diagrams 1
SQL Creation, Update, Delete, Order by, Logical Operators 4
Inner Joins, Left Join, Group by, Having, Nested Querry 5
Views, Transactions, Triggers 7

Crows Feet ERD diagrams

Store Entity: The system tracks the StoreID, city, state, and store manager ID, e.g., EmpID.
Employee Entity: Employee information consists of an EmpID, first name, middle initial, last name, hire date, monthly salary, job title, and their manager's ID.
Customer Entity : Customer data tracked in the database include CustID, customer name, address (street, city, state, zip), area code, phone, sales rep ID, and credit limit.
Order Entity: The database tracks the order no, order date, customer ID, and the calculated order total amount.
Line Item Entity: In addition to the composite identifier (OrderNo and ProdID), the Line Item Entity stores the order quantity, ship date (for online order), and calculated item total amount.
Product Entity: The product record consists of product ID, description, and unit price.
Shipment Entity: Shipment record include ShipNo, ProdID, CustID, OrderNo, ship date, and shipment qty.
Vendor Entity: A vendor record has vendor ID, vendor name, and one or multiple locations.
Receiving Entity: receiving quantity for each product and receiving date are additional attributes to be captured in the database.



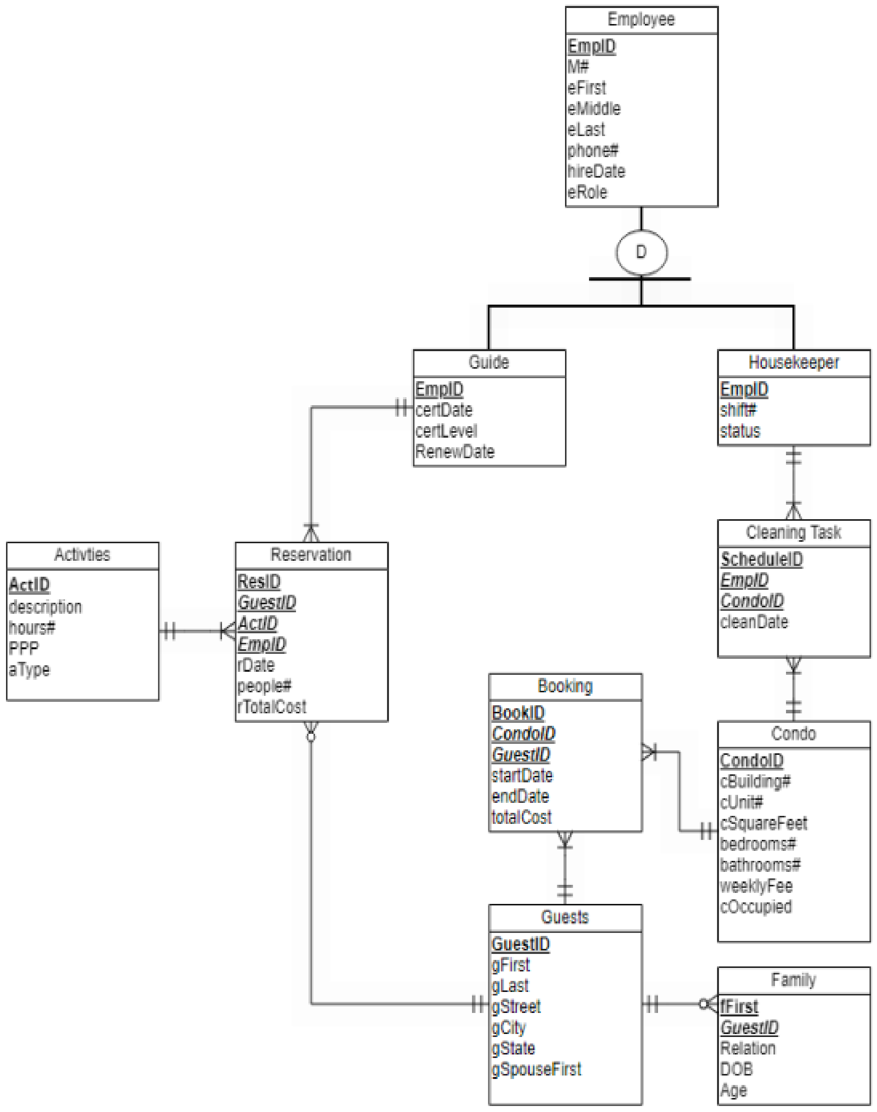
Resort Employee: This database keeps track of employees with different responsibilities, such as managers, front desk clerks, technology support, guides, and housekeepers. All employees have a Employee ID (EmpID), name (last name, middle name, first name), contact phone number, hire date, his/her manager's ID. The management team also wants to record the following additional information for guides and housekeepers. A guide can't be a housekeeper, and vice versa. Guides must be certified every two years to lead the group. In addition to employee ID, each guide has a certified date, certificate level (level 1, level 2, and level 3), and renewal date. Each housekeeper is in one of three shifts (shift 1, shift 2, and shift 3). Some housekeepers are only hired as a temp during the peak season. The database tracks the housekeeper's shift and status (i.e., perm or temp).

Condo and Cleaning: Similar to a hotel room, each condo has a CondoID. The database table also includes the building number, unit number, square footage, number of bedrooms, number of bathrooms, and weeklyfee. During low season, not all condos are occupied. Each condo is assigned to one housekeeper for cleaning, and each housekeeper is assigned to clean multiple condos. When the cleaning task is completed at the end of the shift, each housekeeper logs into the system to enter his/her employee ID, the date, and the condo IDs. The system automatically creates a unique schedule ID for each condo as the record identifier.

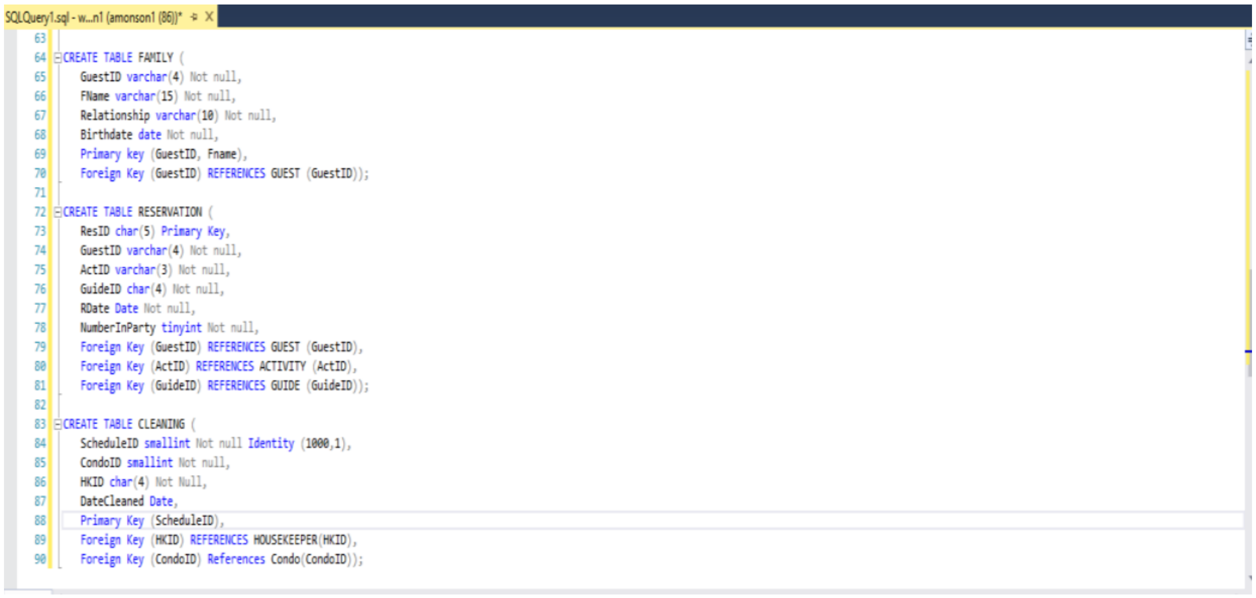
Guests: The resort keeps the last name, first name, address (street, city, state), and spouse's first name of the guest who checked in. Each guest is given a unique VIP membership number (GuestID). The resort also keeps the first name, relationship, birth date, and age of the guest's family members for birthday surprises. (Assuming no two family members have the same first name) Due to its high-quality of service, most of the guests are frequent visitors. The resort also allows guests without family members to join the membership.

Booking: Guests pay for a full week, even if they stay only a few days (condo weekly fee). For example, a 17-day stay will be charged for 3 weeks. The Condo Stay Booking records the booking number (BookID), condo ID, guest ID, start date, and end date. Total booking amount is calculated based on the number of days stayed in the condo and the weekly fee.

Activities and Reservations: The resort provides many guided activities for members to enjoy. These activities require a reservation to assign a date and a guide for the tour. Activities requiring a reservation are stored in the database with a unique ID (ActID) along with description, number of hours, price per person (PPP), and type (hiking, horseback riding, etc.). A guest can reserve many different family activities or none during the stay. Each reservation is given a unique ID (ResID) for billing purpose. The reservation also records the guest ID, activity ID, guide ID, reserved date, number of people in the party, and the total cost. Each reservation requires one and only one guide to lead the group. Because the activity fee is charged per person, the total cost for each reservation is PPP x number of people in the party.

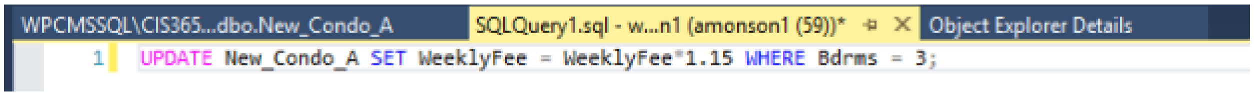

Employee
EmpID
M#
eFirst
eMiddle
eLast
phone#
hireDate
eRole
D
Guide
EmpID
certDate
certLevel
RenewDate
Housekeeper
EmpID
shift#
status
Cleaning Task
ScheduleID
EmpID
CondoID
cleanDate
Activties
ActID
description
hours#
PPP
aType
Reservation
ResID
GuestID
ActID
EmpID
rDate
people#
rTotalCost
Booking
BookID
CondoID
GuestID
startDate
endDate
totalCost
Condo
CondoID
cBuilding#
cUnit#
cSquareFeet
bedrooms#
bathrooms#
weeklyFee
cOccupied
Guests
GuestID
gFirst
gLast
gStreet
gCity
gState
gSpouseFirst
Family
fFirst
GuestID
Relation
DOB
Age

SQL Creation, Update, Delete, Order by, Logical Operators


SQLQuery1.sql - w...n1 (amonson1 (86))*

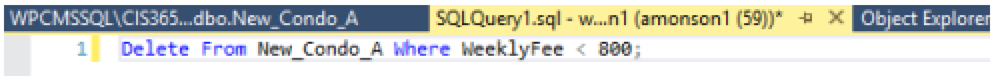
```sql

CREATE TABLE FAMILY (
    GuestID varchar(4) Not null,
    FName varchar(15) Not null,
    Relationship varchar(10) Not null,
    Birthdate date Not null,
    Primary key (GuestID, Fname),
    Foreign Key (GuestID) REFERENCES GUEST (GuestID));

CREATE TABLE RESERVATION (
    ResID char(5) Primary Key,
    GuestID varchar(4) Not null,
    ActID varchar(3) Not null,
    GuideID char(4) Not null,
    RDate Date Not null,
    NumberInParty tinyint Not null,
    Foreign Key (GuestID) REFERENCES GUEST (GuestID),
    Foreign Key (ActID) REFERENCES ACTIVITY (ActID),
    Foreign Key (GuideID) REFERENCES GUIDE (GuideID));

CREATE TABLE CLEANING (
    ScheduleID smallint Not null Identity (1000,1),
    CondoID smallint Not null,
    HKID char(4) Not Null,
    DateCleaned Date,
    Primary Key (ScheduleID),
    Foreign Key (HKID) REFERENCES HOUSEKEEPER(HKID),
    Foreign Key (CondoID) References Condo(CondoID));
```

Write a query to update records in New_Condo_A by adding 15% to the weeklyfee for all 3 bedrooms condos. Display the updated result.


WPCMSSQL\CIS365...dbo.New_Condo_A | SQLQuery1.sql - w...n1 (amonson1 (59))* | Object Explorer Details

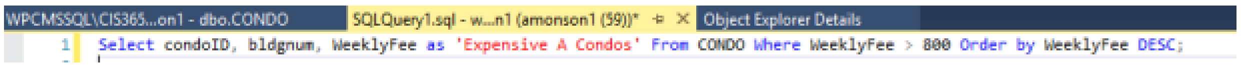
```sql
UPDATE New_Condo_A SET WeeklyFee = WeeklyFee*1.15 WHERE Bdrms = 3;
```

Based on the result of Q1, write a query to delete records in New_Condo_A having the weeklyfee less than $800. Display all records left in New_Condo_A.


WPCMSSQL\CIS365...dbo.New_Condo_A | SQLQuery1.sql - w...n1 (amonson1 (59))* | Object Explorer

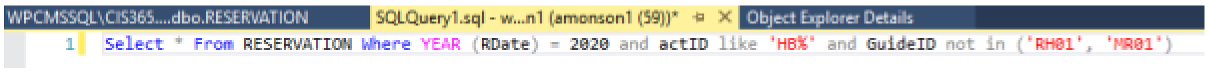
```sql
Delete From New_Condo_A Where WeeklyFee < 800;
```

Use the FamCation CONDO table to answer this question. Write a query to show all the condos in building A that costs more than $800 per week. Display Condo ID, Building Number, and Weekly Fee. List the most expensive condo first. Change the column name of WeeklyFee to 'Expensive A Condos'.


WPCMSSQL\CIS365...on1 - dbo.CONDO | SQLQuery1.sql - w...n1 (amonson1 (59))* | Object Explorer Details

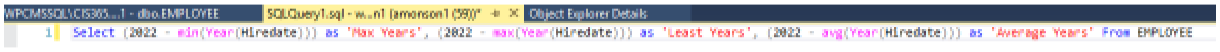
```sql
Select condoID, bldgnum, WeeklyFee as 'Expensive A Condos' From CONDO Where WeeklyFee > 800 Order by WeeklyFee DESC;
```

Write a query to show all the reservations made in 2020 with an activity ID starting with 'HB' that were not led by guide RH01 or MR01


WPCMSSQL\CIS365....dbo.RESERVATION
SQLQuery1.sql - w...n1 (amonson1 (59))*
Object Explorer Details

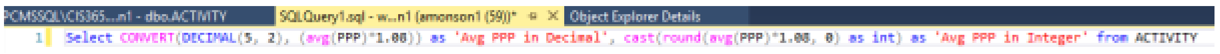
```sql
Select * From RESERVATION Where YEAR (RDate) = 2020 and actID like 'HB%' and GuideID not in ('RH01', 'MR01')
```

Display the most number of years, the least number of years, and the average number of years FamCation Resort employees have been with the company? Label the columns as 'Most Years', 'Least Years', and 'Average Years'.


WPCMSSQL\CIS365...1 - dbo.EMPLOYEE
SQLQuery1.sql - w...n1 (amonson1 (59))*
Object Explorer Details

```sql
Select (2022 - min(Year(Hiredate))) as 'Max Years', (2022 - max(Year(Hiredate))) as 'Least Years', (2022 - avg(Year(Hiredate))) as 'Average Years' From EMPLOYEE
```

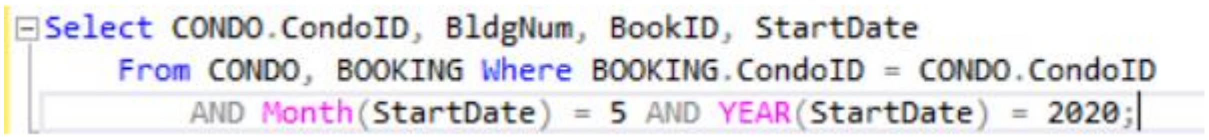
Write a select statement to display the increase of average PPP by 8% for all activities in two columns (Do not actually update of data). The first column shows the result with two significant digits, and the second column shows a rounded full integer. Label these two columns, Avg PPP in Decimal and Avg PPP in Integer, respectively


PCMSSQL\CIS365...n1 - dbo.ACTIVITY
SQLQuery1.sql - w...n1 (amonson1 (59))*
Object Explorer Details

```sql
Select CONVERT(DECIMAL(5, 2), (avg(PPP)*1.08)) as 'Avg PPP in Decimal', cast(round(avg(PPP)*1.08, 0) as int) as 'Avg PPP in Integer' from ACTIVITY
```

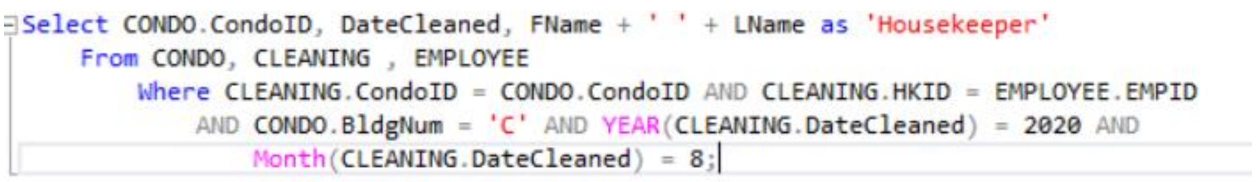
Inner Joins, Left Join, Group by, Having, Nested Querry

Display Condo id, building number, booking id, and booking start date in May of 2020.



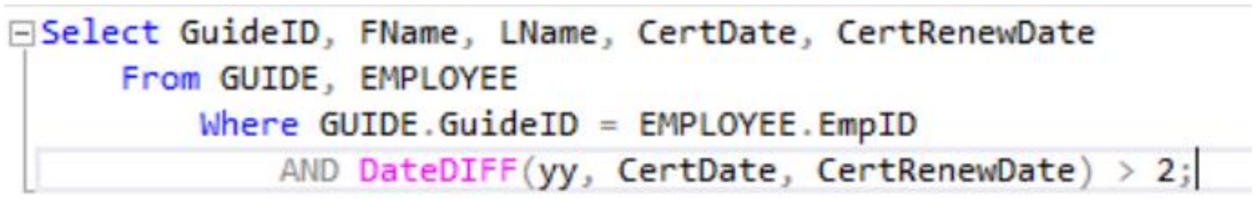
```sql
Select CONDO.CondoID, BldgNum, BookID, StartDate
    From CONDO, BOOKING Where BOOKING.CondoID = CONDO.CondoID
        AND Month(StartDate) = 5 AND YEAR(StartDate) = 2020;
```

Show all building C condos cleaned in August 2020. Display the condo ID, date cleaned, housekeeper's first and last name. Display and label the first name and last name together in one column as Housekeeper



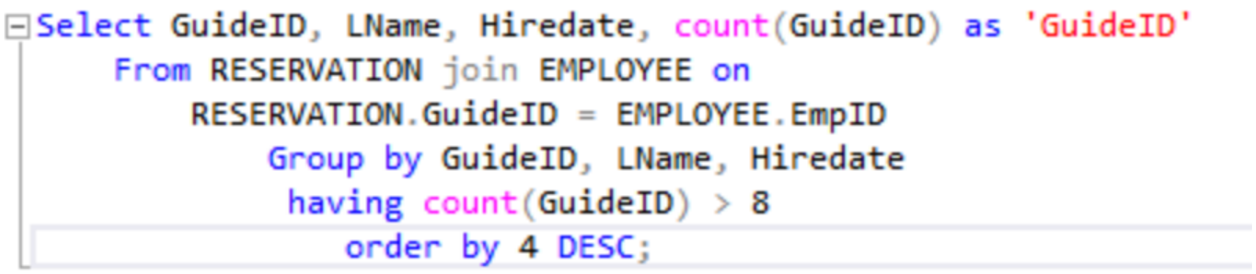
```sql
Select CONDO.CondoID, DateCleaned, FName + ' ' + LName as 'Housekeeper'
    From CONDO, CLEANING , EMPLOYEE
        Where CLEANING.CondoID = CONDO.CondoID AND CLEANING.HKID = EMPLOYEE.EMPID
            AND CONDO.BldgNum = 'C' AND YEAR(CLEANING.DateCleaned) = 2020 AND
                Month(CLEANING.DateCleaned) = 8;
```

Display guides who didn't renew their certificates within 2 years. List their ID, first name, last name, certificate date, and renew date



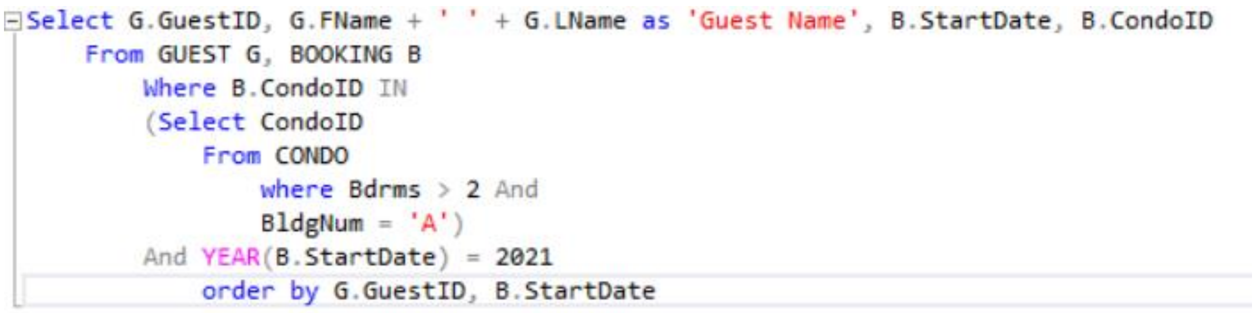
```
Select GuideID, FName, LName, CertDate, CertRenewDate
    From GUIDE, EMPLOYEE
        Where GUIDE.GuideID = EMPLOYEE.EmpID
            AND DateDIFF(yy, CertDate, CertRenewDate) > 2;
```

Each guide can lead multiple reservations. Show guides who have led more than 8 reservations. Display guide id, last name, hire date, the number of reservations (label as "Reservation per Guide"). The highest count will be displayed first. Use group by, having, and order by



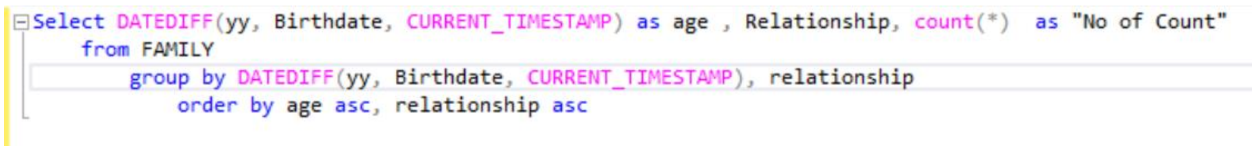
```
Select GuideID, LName, Hiredate, count(GuideID) as 'GuideID'
    From RESERVATION join EMPLOYEE on
        RESERVATION.GuideID = EMPLOYEE.EmpID
            Group by GuideID, LName, Hiredate
             having count(GuideID) > 8
                order by 4 DESC;
```

Write a sub-query for condoID to list out all the guests booked a 3-bedroom building A condo in 2021. Display the guest id, guest first name and last name, booking start date, and condo id. Label the 'Guest Name' and order the result by guest ID and then start date.



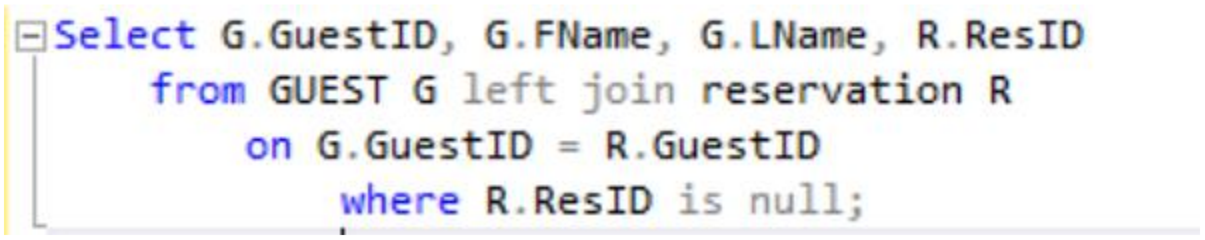
```
Select G.GuestID, G.FName + ' ' + G.LName as 'Guest Name', B.StartDate, B.CondoID
    From GUEST G, BOOKING B
        Where B.CondoID IN
        (Select CondoID
            From CONDO
                where Bdrms > 2 And
                BldgNum = 'A')
        And YEAR(B.StartDate) = 2021
            order by G.GuestID, B.StartDate
```

Count the number of girls and boys in each age group. The result will show age group, relationship, and count number. Order the result by age from the youngest to the oldest, and then by relationship. Do not use UNION in your query.



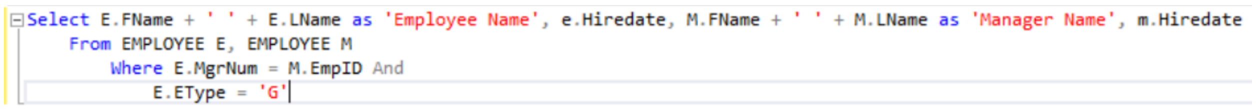
```
Select DATEDIFF(yy, Birthdate, CURRENT_TIMESTAMP) as age , Relationship, count(*)  as "No of Count"
    from FAMILY
        group by DATEDIFF(yy, Birthdate, CURRENT_TIMESTAMP), relationship
            order by age asc, relationship asc
```

Use left-join to show all the guests who didn't reserve any activities. The result will show the guest ID, firstname, last name, and reservation ID (should be Null).



```
Select G.GuestID, G.FName, G.LName, R.ResID
    from GUEST G left join reservation R
        on G.GuestID = R.GuestID
            where R.ResID is null;
```

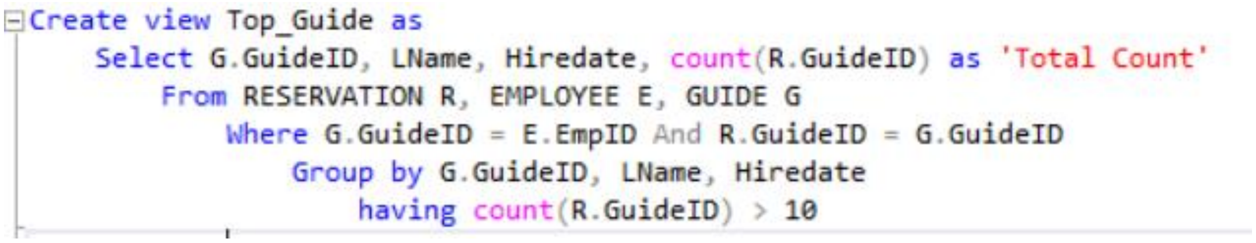
Use self-join to show all the guide's full name, hired date, their manager's full name, manager's hired date.



```
Select E.FName + ' ' + E.LName as 'Employee Name', e.Hiredate, M.FName + ' ' + M.LName as 'Manager Name', m.Hiredate
    From EMPLOYEE E, EMPLOYEE M
        Where E.MgrNum = M.EmpID And
            E.EType = 'G'
```

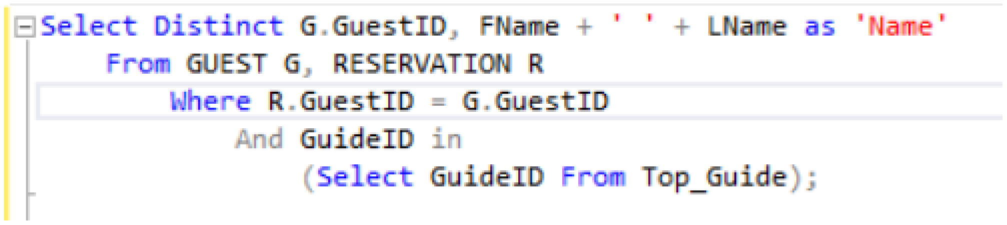
Views, Transactions, Triggers

Create a view named 'Top_Guide' listing guides who have led more than 10 reservations. The view will displayguide ID, last name, hired date, and total count of reservations.



```
Create view Top_Guide as
    Select G.GuideID, LName, Hiredate, count(R.GuideID) as 'Total Count'
        From RESERVATION R, EMPLOYEE E, GUIDE G
            Where G.GuideID = E.EmpID And R.GuideID = G.GuideID
                Group by G.GuideID, LName, Hiredate
                    having count(R.GuideID) > 10
```

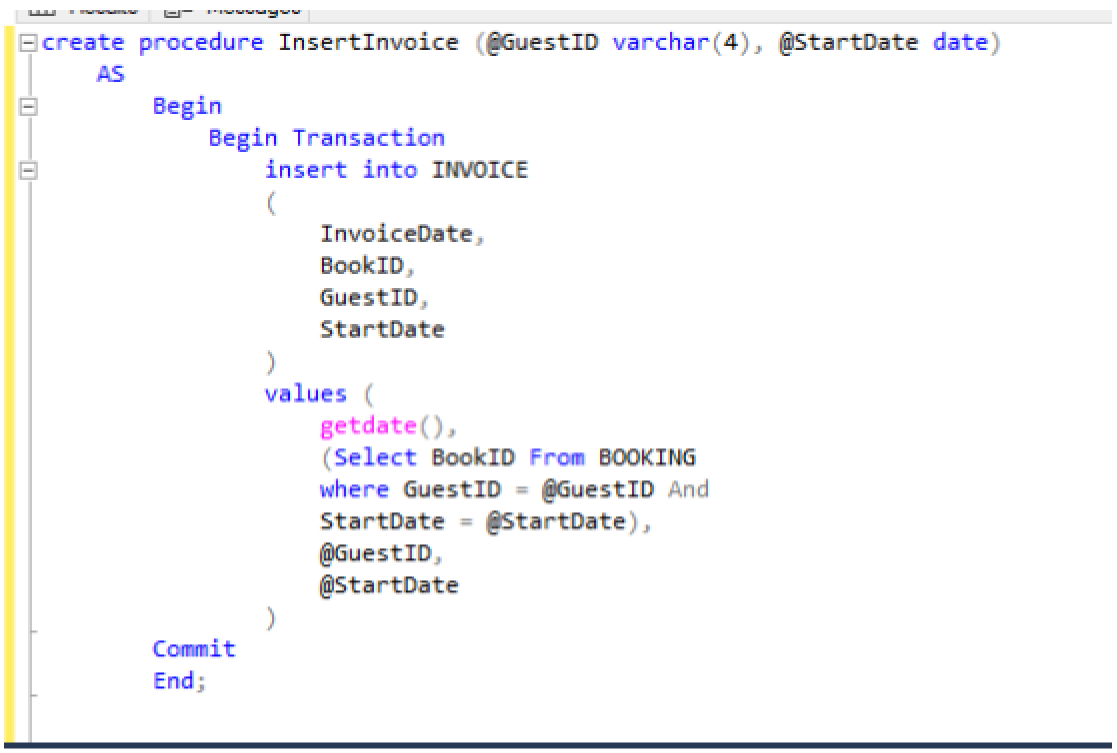
Use the created view to list all guests (first name and last name) who have made reservations with these topguides via a 'noncorrelated' subquery. Don't show duplicate guest names.



```
Select Distinct G.GuestID, FName + ' ' + LName as 'Name'
    From GUEST G, RESERVATION R
        Where R.GuestID = G.GuestID
            And GuideID in
                (Select GuideID From Top_Guide);
```

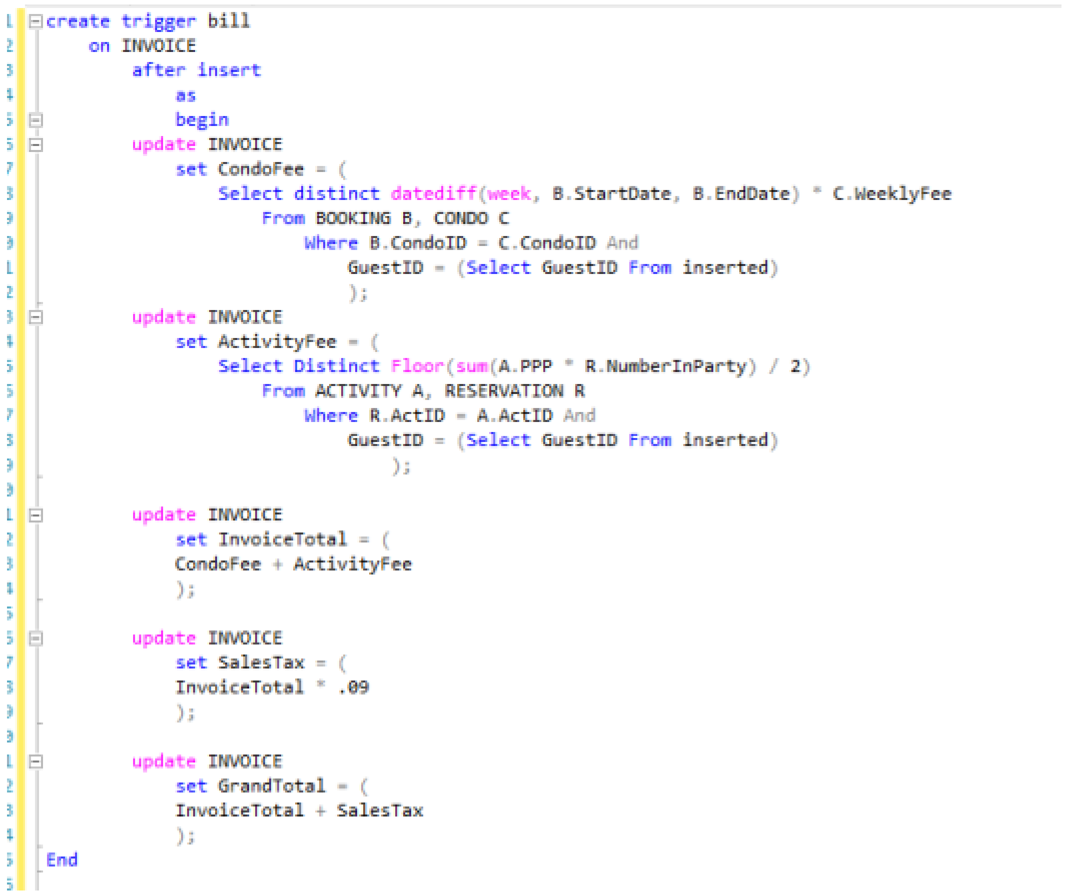
Create a stored procedure to insert a record into INVOICE table using the input parameters passed.
Name the stored procedure as: InsertInvoice
The stored procedure will have the following steps:

1) Procedure should accept two input parameters: GuestID and Startdate
2) Begin the transaction
3) Insert values to the first 5 attributes of the INVOICE table
1. InvoiceID: Identity constraint will automatically create an invoice number for each insertion.
2. InvoiceDate: auto-default to today's date
3. Booking ID: select bookid from the booking table based on the GuestID and StartDate passed as input parameters.
4. GuestID: use the GuestID passed passed as input parameter
5. StartDate: use the StartDate passed as input parameter
4) Commit the transaction



```
create procedure InsertInvoice (@GuestID varchar(4), @StartDate date)
    AS
        Begin
            Begin Transaction
                insert into INVOICE
                (
                    InvoiceDate,
                    BookID,
                    GuestID,
                    StartDate
                )
                values (
                    getdate(),
                    (Select BookID From BOOKING
                    where GuestID = @GuestID And
                    StartDate = @StartDate),
                    @GuestID,
                    @StartDate
                )
        Commit
        End;
```

Create an 'After Insert' Trigger on the INVOICE table.
Whenever a record is inserted into INVOICE table, this trigger will automatically calculate the invoice related columns and update INVOICE table.
"Current record" will always be available in the special table called INSERTED
Assume that the following columns of INVOICE table will already be populated by an INSERT statement for the "current record":
InvoiceID, InvoiceDate, BookID, GuestID, StartDate



```
create trigger bill
    on INVOICE
        after insert
            as
            begin
        update INVOICE
            set CondoFee = (
                Select distinct datediff(week, B.StartDate, B.EndDate) * C.WeeklyFee
                    From BOOKING B, CONDO C
                        Where B.CondoID = C.CondoID And
                            GuestID = (Select GuestID From inserted)
                            );
        update INVOICE
            set ActivityFee = (
                Select Distinct Floor(sum(A.PPP * R.NumberInParty) / 2)
                    From ACTIVITY A, RESERVATION R
                        Where R.ActID = A.ActID And
                            GuestID = (Select GuestID From inserted)
                                );

        update INVOICE
            set InvoiceTotal = (
            CondoFee + ActivityFee
            );

        update INVOICE
            set SalesTax = (
            InvoiceTotal * .09
            );

        update INVOICE
            set GrandTotal = (
            InvoiceTotal + SalesTax
            );
End
```