Manipulera data i befintlig databas med hjälp av SQL

Vi vill skapa nya tabeller från befintlig data i databasen ”everyloop”, och sedan manipulera och ta ut data från dessa nya tabeller med hjälp av SQL.

Lösningar till laborationen lämnas in i i form av ett SQL-script som steg för steg löser uppgifterna i den ordning som de finns beskrivna nedan.

Varje uppgift ska tydligt separeras från den föregående med en lämplig radkommentar (exempelvis hellinje). Lägg dessutom till en radkommentar som anger vilken den aktuella uppgiften är uppgift.

Uppgifter – Tabell : *MoonMissions*

1. Använd ”SELECT INTO” för att ta ut kolumnerna ’Spacecraft’, ’Launch date’, ’Carrier rocket’, ’Operator’, samt ’Mission type’ för alla lyckade uppdrag (Successful outcome) och sätt in i en ny tabell med namn ”SuccessfulMissions”.

2. I kolumnen ’Operator’ har det smugit sig in ett eller flera mellanslag före operatörens namn skriv en query som uppdaterar ”SuccessfulMissions” och tar bort mellanslagen kring operatör.

3. Skriv en select query som tar ut, grupperar, samt sorterar på kolumnerna ’Operator’ och ’Mission type’ från ”SuccessfulMissions”. Som en tredje kolumn ’Mission count’ i resultatet vill vi ha antal uppdrag av varje operatör och typ. Ta bara med de grupper som har fler än ett (>1) uppdrag av samma typ och operatör.

4. För betyg VG (endast följande uppgift):
I ett flertal fall har värden i kolumnen ’Spacecraft’ ett alternativt namn som står inom parantes, t.ex: ”Pioneer 0 (Able I)”. Skriv en query som uppdaterar ”SuccessfulMissions” på ett sådant sätt att alla värden i kolumnen ’Spacecraft’ endast innehåller originalnamnet, dvs ta bort alla paranteser och deras innehåll. Ex: ”Pioneer 0 (Able I)” => ”Pioneer 0”.

Uppgifter – Tabell: *Users*

5. Ta ut samtliga rader och kolumner från tabellen ”Users”, men slå ihop ’Firstname’ och ’Lastname’ till en ny kolumn ’Name’, samt lägg till en extra kolumn ’Gender’ som du ger värdet ’Female’ för alla användare vars näst sista siffra i personnumret är jämn, och värdet ’Male’ för de användare där siffran är udda. Sätt in resultatet i en ny tabell ”NewUsers”.

6. Skriv en query som returnerar en tabell med alla användarnamn i ”NewUsers” som inte är unika i den första kolumnen, och antalet gånger de är duplicerade i den andra kolumnen.

7. Skriv en följd av queries som uppdaterar de användare med dubblerade användarnamn som du fann ovan, så att alla användare får ett unikt användarnamn. D.v.s du kan hitta på nya användarnamn för de användarna, så länge du ser till att alla i ”NewUsers” har unika värden på ’Username’.

8. Skapa en query som tar bort alla kvinnor födda före 1970 från ”NewUsers”.

9. Lägg till en ny användare (hitta på en) i tabellen ”NewUsers”.

10. För betyg VG (endast följande uppgift):
Skriv en query som returnerar två kolumner ’gender’ och ’avarage age’, och två rader där ena raden visar medelåldern för män, och andra raden visar medelåldern på kvinnor för alla användare i tabellen ”NewUsers”.

Uppgifter – Tabeller: *Company* (Joins)

11. Skriv en query som selectar ut alla (77) produkter i company.products

Dessa ska visas i 4 kolumner:

Id – produktens id

Product – produktens namn

Supplier – namnet på företaget som leverar produkten

Category – namnet på kategorin som produkten tillhör

12. Skriv en query som listar antal anställda i var och en av de fyra regionerna i tabellen company.regions

13. För betyg VG (endast följande uppgift):

Vi har tidigare kollat på one-to-many och many-to-many joins. Det finns även det som brukar kallas för self-join, när en tabell joinar mot sig själv.

Sök & läs på om self-joins!

Använd en self-join för att lista alla (9) anställda och deras närmsta chef.

De anställda ska visas i tre kolumner:

Id – Den anställdes id.

Name – Den anställdes titel och fullständiga namn (ex: Dr. Andrew Fuller)

Reports to – Närmsta chefens titel och fullständiga namn.

I de fall ReportsTo-kolumnen i company.employer är NULL, visa ’Nobody!’

Redovisning

Labben ska lösas individuellt.

Skriv queries som löser problemen ovan och spara i en .sql fil som du döper till ditt för- och efternamn. Exempel: ”FredrikJohansson.sql”.

Ladda upp filen för inlämning direkt på ITHSdistans.se

Betygskriterier

För godkänt krävs:

• SQL-scriptet ska lämnas in på ITHS.
• SQL-koden ska vara formaterad i ett lättläsligt format.
• Queries ska i tur och ordning lösa uppgifterna i laborationen.
• Alla delmoment förutom de 3 som är märkta ”För betyg väl godkänt” ska vara korrekt lösta.

Glöm ej att avsluta varje query/commando med ett semicolon!

För väl godkänt krävs även:

• Alla delmoment inklusive de 3 som är märkta ”För betyg VG” ska vara