Teoretiska Frågor

Beskriv kort hur en relationsdatabas fungerar.

En relationsdatabas består av ”tables” eller ”tabeller”, som består av rader och kolumner. Namnet kommer från att dessa tabeller har relationer mellan sig, och med hjälp av ”nycklar” kan man identifiera dessa relationer och t.ex. hämta/jämföra data ur flera relaterade tabeller samtidigt. Med en relationsdatabas undviker man att ha en enda stor tabell, vilket t.ex. gör bearbetning mycket smidigare.

Vad menas med ”CRUD” flödet?

CRUD står för Create, Read, Update och Delete, och är vanligt inom mjukvaru-utveckling. Beroende på om det är SQL eller något annat språk, så kan andra ord användas -men betyda samma sak. I SQL används CREATE (och INSERT) till C i CRUD, SELECT till R, UPDATE (och SET) till U, och DELETE till D. Med dessa har man kontroll över sin databas via språket, istället för att behöva använda GUI, och kan alltså skapa tables, hämta data, uppdatera celler, och även radera, om man har den befogenheten.

Beskriv kort vad en ”left join” och ”inner join” är. Varför använder man det?

Joins är när man hämtar från flera tabeller samtidigt, via nycklar. Detta är för att man t.ex. vill se och jämföra data mellan dom. Om man t.ex. har en table med produkter, och en table med försäljningssiffror, så kommer en INNER JOIN visa de rader som har matchande värde, alltså de produkter som faktiskt har sålt. Med en LEFT JOIN, och produkter som den först nämnda tabellen i koden (left), kommer man kunna se alla produkter, även de som inte har försäljningsdata och kanske visar null i de cellerna som inte följde med i en INNER JOIN. Man väljer typ av join beroende på om man vill se alla produkter för att få den överblicken, eller om man vill man se enbart de som sålt.

Beskriv kort vad indexering i SQL innebär.

En indexering kan skapas för att kunna söka av tabeller snabbare. Man kan använda CREATE / DROP INDEX för att skapa/ta bort en indexering från en tabell. En anledning till att man inte har indexerat alla tabeller, är att det tar längre tid att uppdatera tabeller, för själva indexeringen behöver då också uppdateras. Det är därför förmodligen bäst att indexera tabeller som mest bara används för sökning och inte uppdattering.

Beskriv kort vad en vy i SQL är.

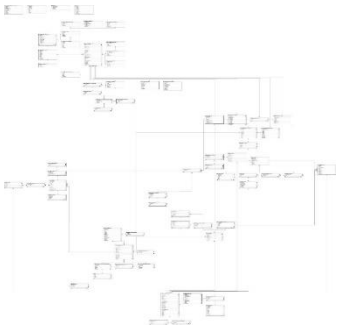
Om det finns en sannolikhet att man (eller någon annan användare) kommer vilja använda t.ex. en mer avancerad Query man skrivit, vars resultat är en tabell, så kan man spara den som en View. Denna View visar resultatet av Queryn, som en sparad ”virtuell” tabell som alltså inte är en av de faktiska tabellerna i databasen. Denna är dynamisk, vilket betyder att om datan från Viewn är från en annan tabell, som nu får sin data uppdatterad, så ändras också resultatet i Viewn.

Beskriv kort vad en lagrad procedur i SQL är.

Med en lagrad procedur, kan man med CREATE PROCEDURE i princip spara en query i en variabel, och sedan köra variabeln endast, utan att behöva skriva om hela queryn, med kommandot EXEC. Det är ett smidigt sätt att minska upprepning och mängden kod som tar upp yta. Det är möjligt att skicka med argument in i proceduren, så att den blir ännu mer användbar, likt parametrar i en funktion.

En deskriptiv sammanfattning över AdventureWorks2022

För att få en överblick av databasen ur ett annat perspektiv, skapade jag ett databas-diagram genom att högerklicka på ”Database Diagrams” och sedan inkludera alla Tables. Väldigt utzoomad bild nedan.



Efter att ha organiserat tabellerna, ser man direkt några olika intressanta saker. Ett par tables har inga relationer alls till andra tables, t.ex. ErrorLog, AWBuildVersion och DatabaseLog. De är helt enkelt avskilda från resten av alla tables i att de inte har någon relation med all annan data. De verkar mer vara till för hantering av själva databasen. DatabaseLog verkar vara till för att hålla ordning på olika förändringar i databasen som gjorts med t.ex. CREATE och ALTER.

TransactionHistoryArchive, versionen av TransactionHistory som är byggd för att arkivera mängden transaktioner, står också utanför relationerna enlight diagramet, men ändå inenhåller de samma typ av data. Skillnaden är att TransactionHistoryArchive inte har fått tilldelat t.ex. ProductID som foreign key, vilket TransactionHistory använder för att kunna kopplas till Product-tablen.

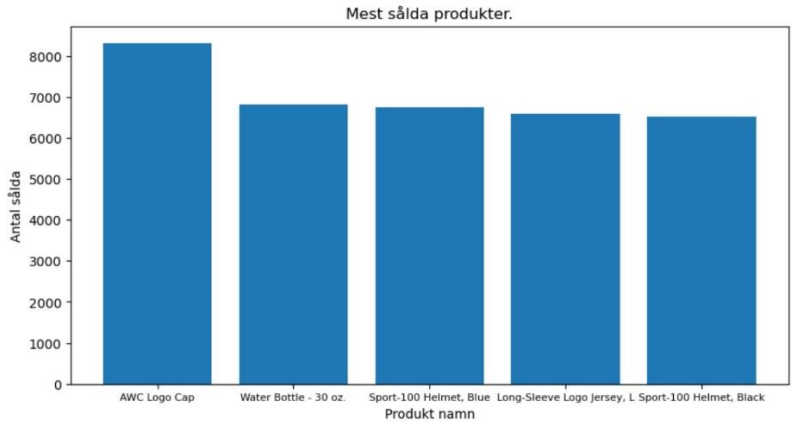
Vissa tables, t.ex. Product i Production-schemat agerar nästan som en hubb med många tables runt omkring som har relation till just Product, t.ex. ProductSubcategory, ProductModel, och TransactionHistory. Många av dessa relationer är inom schemat Production, vilket kanske inte är så förvånande. Sedan finns även färre relations-kopplingar till andra schemas, t.ex. Purchasing, och vidare till Sales, Person.

Produktion, inköpt och försäljning verkar vara en del av den här databasen, och då även i företaget. I ProductModel finns väldigt mycket cykel-relaterat t.ex. cykel-hjul, styren, och även vattenflaskor. Med produkt-namn som Mountain Bike Socks, så kan man ana att det är en kombination av cyklar och cykel-relaterade föremål, med äventyr/outdoor. Det förklarar isåfall databasens namn ”AdventureWorks”.I table Location kan man även se att företaget arbetar med Frame Welding ända till Final Assembly. BillOfMaterials är t.ex. inte förvånande kopplat till Production då material krävs för tillverkning.

```
SELECT TOP 10 p.Name, SUM(sod.OrderQty) AS [Total Orders]
FROM Production.Product p JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID GROUP BY Name
ORDER BY [Total Orders] DESC;
```

	Name	Total Orders
1	AWC Logo Cap	8311
2	Water Bottle - 30 oz.	6815
3	Sport-100 Helmet, Blue	6743
4	Long-Sleeve Logo Jersey, L	6592
5	Sport-100 Helmet, Black	6532
6	Sport-100 Helmet, Red	6266
7	Classic Vest, S	4247
8	Patch Kit/8 Patches	3865
9	Short-Sleeve Classic Jersey, XL	3864
10	Long-Sleeve Logo Jersey, M	3636

Här tänkte jag vi kunde titta på vilka produkter som säljer mest, via Product och SalesOrderDetail-tabellerna, samt nyckeln ProductID, och det är en keps. Eftersom den antagligen är ganska billig jämfört med andra produkter, så kan vi också titta på vad som sålt mest genom att modifiera koden lite genom att skapa en ny kolumn för total price, och sortera därefter, och det är cykeln Mountain-200 Black, 38.




```
SELECT TOP 10 p.Name, SUM(sod.OrderQty) AS [Total Orders],
sum(sod.UnitPrice * sod.OrderQty) AS [Total Price]
FROM Production.Product p JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID GROUP BY Name
ORDER BY [Total Price] DESC;
```

	Name	Total Orders	Total Price
1	Mountain-200 Black, 38	2977	4406151,2662
2	Mountain-200 Black, 42	2664	4014067,7999
3	Mountain-200 Silver, 38	2394	3696486,4726
4	Mountain-200 Silver, 42	2234	3441292,5443
5	Mountain-200 Silver, 46	2216	3436090,7946
6	Mountain-200 Black, 46	2111	3311098,4385
7	Road-250 Black, 44	1642	2518299,7641
8	Road-250 Black, 48	1498	2348246,0923
9	Road-250 Black, 52	1245	2012447,775
10	Road-150 Red, 56	664	1847818,628

I Purchasing schemat finns Vendors kopplade till produktionen, vilket betyder att materialet köps in till tillverkning. Här finns information om Vendors, kostnad, orderDate m.m. PurchaseOrderHeader är också kopplat till SalesOrderHeader med endast ShipMethod som brygga, vilket tyder på att A.W kan köpa in och sälja direkt, som en mellanhand utan att gå via tillverkning.

I Sales finns SalesPerson med information om t.ex. territoryID, commision m.m. Här finns också SalesorderHeader med information om allt från OrdetDate till AccountNumber och status. Inom Sales schemat finns både Store och Customer, där SalesPerson är kopplad till Store via BuisnessEntityID, och till Customer via TerritoryID och table SalesTerritoryHistory. Det verkar alltså som att A.W säljer både till affärer och till enskilda kunder.

I databasen finns också schemas över Human Resources, med information om olika departement, och information om individerna som är anställda där, till och med ”marital status” (vilket förmodligen används när mail och brev skickas).

Human Resources, och andra schemas, är föga förvånande kopplat till bland annat schemat Person, som bland annat har information om VacationHours, Gender, JobTitle m.m. Här finns andra tables inom samma schema, med bland annat telefonnummer och lösenord.

Jag blev plötsligt nyfiken på om det finns något man kan se om försäljning och sjukdagar m.m, och för det behöver jag tre tabeller så att jag får med allt jag behöver, info om personen från HumanResources-schemat, försäljning från SalesPerson, och namn från Person.

```
SELECT TOP 15 e.Gender, e.VacationHours, e.SickLeaveHours, p.FirstName, p.LastName, s.Bonus,
s.SalesLastYear FROM HumanResources.Employee AS e join Person.Person AS p ON
E.BusinessEntityID = P.BusinessEntityID JOIN Sales.SalesPerson s ON s.BusinessEntityID =
e.BusinessEntityID ORDER BY s.SalesLastYear DESC;
```

Det verkar ganska jämnt fördelat mellan kvinnor och män. Det ser ut som att Ranjit fått väldigt liten bonus jämfört med sina kollegor. Jag gör en ändring och kommer nu inkludera en ny kolumn som kombinerar detta året med förra årets försäljning, för att kanske få en mer rättvis bild över hur bonus-situationen ser ut.

	Gender	VacationHours	SickLeaveHours	FirstName	LastName	Bonus	SalesLastYear
1	M	34	37	Ranjit	Varkey Chudukatil	985,00	2396539,7601
2	F	36	38	Lynn	Tsoflias	5650,00	2278548,9776
3	M	26	33	Shu	Ito	3550,00	2073505,9999
4	M	31	35	José	Saraiva	5000,00	2038234,6549
5	F	24	32	Jillian	Carson	2500,00	1997186,2037
6	F	22	31	Pamela	Ansman-Wolfe	5000,00	1927059,178
7	M	29	34	Tsvi	Reiter	6700,00	1849640,9418
8	M	38	39	Michael	Blythe	4100,00	1750406,4785
9	F	37	38	Jae	Pak	5150,00	1635823,3967
10	M	33	36	Garrett	Vargas	500,00	1620276,8966
11	F	27	33	Linda	Mitchell	2000,00	1439156,0291
12	M	23	31	David	Campbell	3500,00	1371635,3158
13	F	35	37	Rachel	Valdez	75,00	1307949,7917
14	M	14	27	Stephen	Jiang	0,00	0,00
15	F	21	30	Amy	Alberts	0,00	0,00

SELECT TOP 15 e.Gender, e.VacationHours, e.SickLeaveHours, p.FirstName, p.LastName, s.Bonus, s.SalesLastYear, s.SalesQuota, s.SalesYTD, s.SalesLastYear + s.SalesYTD AS [Total Sales] FROMHumanResources.Employee AS e join Person.Person AS p ON E.BusinessEntityID = P.BusinessEntityID JOIN Sales.SalesPerson s ON s.BusinessEntityID = e.BusinessEntityID ORDER BY [Total Sales] DESC;

	Gender	VacationHours	SickLeaveHours	FirstName	LastName	Bonus	SalesLastYear	SalesQuota	SalesYTD	Total Sales
1	F	37	38	Jae	Pak	5150,00	1635823,3967	250000,00	4116871,2277	5752694,6244
2	F	27	33	Linda	Mitchell	2000,00	1439156,0291	250000,00	4251368,5497	5690524,5788
3	M	34	37	Ranjit	Varkey Chudukatil	985,00	2396539,7601	250000,00	3121616,3202	5518156,0803
4	M	38	39	Michael	Blythe	4100,00	1750406,4785	300000,00	3763178,1787	5513584,6572
5	F	24	32	Jillian	Carson	2500,00	1997186,2037	250000,00	3189418,3662	5186604,5699
6	M	31	35	José	Saraiva	5000,00	2038234,6549	250000,00	2604540,7172	4642775,3721
7	M	26	33	Shu	Ito	3550,00	2073505,9999	250000,00	2458535,6169	4532041,6168
8	M	29	34	Tsvi	Reiter	6700,00	1849640,9418	300000,00	2315185,611	4164826,5528
9	F	36	38	Lynn	Tsoflias	5650,00	2278548,9776	250000,00	1421810,9242	3700359,9018
10	F	22	31	Pamela	Ansman-Wolfe	5000,00	1927059,178	250000,00	1352577,1325	3279636,3105
11	F	35	37	Rachel	Valdez	75,00	1307949,7917	250000,00	1827066,7118	3135016,5035
12	M	33	36	Garrett	Vargas	500,00	1620276,8966	250000,00	1453719,4653	3073996,3619
13	M	23	31	David	Campbell	3500,00	1371635,3158	250000,00	1573012,9383	2944648,2541
14	M	39	39	Tete	Mensa-Annan	3900,00	0,00	300000,00	1576562,1966	1576562,1966
15	M	14	27	Stephen	Jiang	0,00	0,00	NULL	559697,5639	559697,5639

Ranjit har ungefär lika mycket total försäljning som Michael Blythe, men väldigt olika Bonus. Kanske beror det på att Ranjit sålt mindre detta året. Jae Pak har sålt mest, och i en tabell där jag slår samman VacationHours med SickLeaveHours, kan man se att Jae Pak är den tredje personen som är mest frånvarande på jobbet, trots försäljningssiffrorna.

SELECT TOP 5 e.Gender, e.VacationHours, e.SickLeaveHours, e.VacationHours + e.SickLeaveHours AS [Not At Work], p.FirstName, p.LastName, s.Bonus, s.SalesLastYear, s.SalesYTD, s.SalesLastYear + s.SalesYTD AS [Total Sales] FROM HumanResources.Employee AS e join Person.Person AS p ON E.BusinessEntityID = P.BusinessEntityID JOIN Sales.SalesPerson s ON s.BusinessEntityID = e.BusinessEntityID ORDER BY [Not At Work] DESC;

	Gender	VacationHours	SickLeaveHours	Not At Work	FirstName	LastName	Bonus	SalesLastYear	SalesYTD	Total Sales
1	M	39	39	78	Tete	Mensa-Annan	3900,00	0,00	1576562,1966	1576562,1966
2	M	38	39	77	Michael	Blythe	4100,00	1750406,4785	3763178,1787	5513584,6572
3	F	37	38	75	Jae	Pak	5150,00	1635823,3967	4116871,2277	5752694,6244
4	F	36	38	74	Lynn	Tsoflias	5650,00	2278548,9776	1421810,9242	3700359,9018
5	F	35	37	72	Rachel	Valdez	75,00	1307949,7917	1827066,7118	3135016,5035

En statistisk analys, och analys av datan.

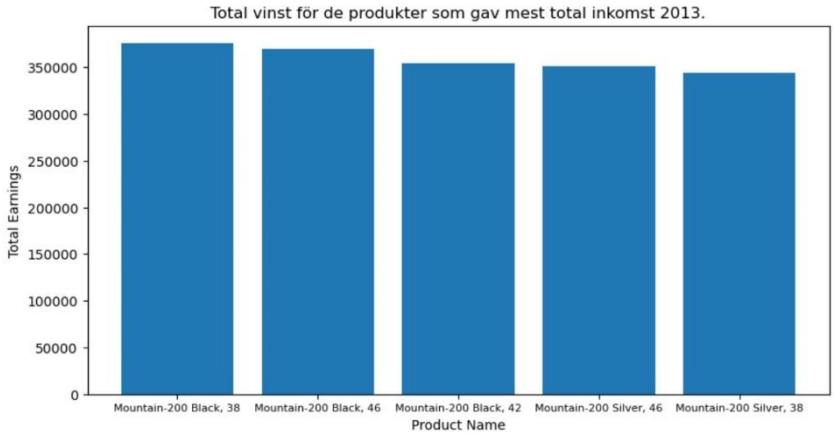
I min statistiska analys tänkte jag titta vidare på de mest sålda varorna, och jämföra med produktionskostnaderna för att se vilken vara som faktiskt är mest lönsam. Jag ska även göra ett konfidensintervall (95%) över hur mycket det mest inkomstbringande varan säljer på en månad.

Jag väljer att titta över senaste fulla året, och genom en sortering efter datum på SalesorderHeader, kan jag se att senaste fulla året är 2013. 2014 har bara inlägg fram till juni.

```
SELECT TOP 5 p.Name, SUM(sod.OrderQty) AS [Total Orders],
        sum(sod.UnitPrice * sod.OrderQty) AS [Total Price],
        sum(p.StandardCost * sod.OrderQty) AS [Total Cost],
        sum(sod.UnitPrice * sod.OrderQty - p.StandardCost * sod.OrderQty) AS [Total Earnings]
        FROM Production.Product p JOIN Sales.SalesOrderDetail sod
        ON p.ProductID = sod.ProductID
        JOIN Sales.SalesOrderHeader soh on soh.SalesOrderID = sod.SalesOrderID
        WHERE soH.OrderDate >= '2013-01-01' AND soh.OrderDate < '2014-01-01'
        GROUP BY Name ORDER BY [Total Earnings] DESC;
```

	Name	Total Orders	Total Price	Total Cost	Total Earnings
1	Mountain-200 Black, 38	1470	2216245,6024	1840412,511	375833,0914
2	Mountain-200 Black, 46	1036	1666952,8643	1297052,6268	369900,2375
3	Mountain-200 Black, 42	1262	1934266,74	1580000,4006	354266,3394
4	Mountain-200 Silver, 46	1033	1658577,4208	1307384,9435	351192,4773
5	Mountain-200 Silver, 38	1164	1817072,5112	1473181,098	343891,4132

Ur datan kan man se att ”Mountain-200 Black, 38” gav mest total vinst under 2013. En intressant sak är att plats två av produkter som ger mest total vinst tas upp av ”BLACK 46”, vilket man inte hade sett om man enbart tittat på ”TOTAL PRICE” - alltså vad den har sålts för. Den hade då hamnat på fjärde plats, men tack vare en mindre total kostnad, så ger den mer total vinst.



Påvägen fann jag något annat intressant. Om man ändrar ORDER BY till [Total Price], så kan man se att nionde plats av de produkter som gett mest total vinst före tillverkningskostan, är ”Road-350-W Yellow, 48”. När man tittar på Total Earnings-kolumnen kan man se att produkten gett en förlust på över 14000 tack vare högre produktionskostnader.

Behåller vi ORDER BY [Total Earnings], men ändrar till ASC, så ser vi flera produkter som företaget förlorar pengar på, varav en som gått back cirka 120´000 2013.

Jag föreslår att tillverkningskostnaderna bör ses över, särskilt gällande produkten/produkterna som har högre tillverkningskostnad än vad den ger i vinst. Är det en produkt som ger någon annan slags vinst, fyller den ett reklamsyfte? En rekommendation kan annars vara att sluta sälja den, höja priset, eller att minska volymen som går att köpa.

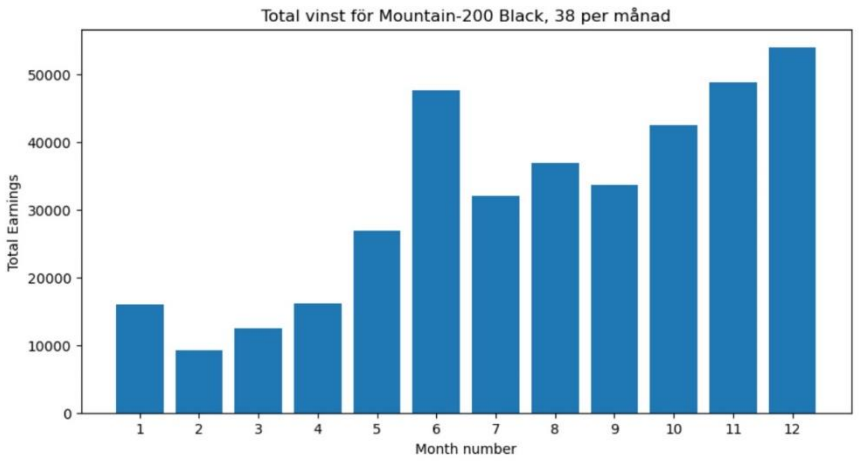
	Name	Total Orders	Total Price	Total Cost	Total Earnings
1	Touring-1000 Yellow, 60	717	942780,4815	1062549,4743	-119768,9928
2	Touring-1000 Yellow, 46	629	846821,664	932138,9391	-85317,2751
3	Touring-3000 Yellow, 62	536	218592,381	247334,4128	-28742,0318
4	Touring-3000 Yellow, 44	546	224211,9705	251948,8608	-27736,8903
5	Touring-3000 Blue, 50	555	229000,128	256101,864	-27101,736

Här följer en analys av hur den mest inkomstbringande varan säljer månadsvis år 2013, i total inkomst.

```
SELECT p.Name, SUM(sod.OrderQty) AS Quantity,
        sum(sod.UnitPrice * sod.OrderQty - p.StandardCost
        * sod.OrderQty) AS [Total Earnings],
        MONTH(soh.[OrderDate]) AS MonthNum FROM
        Production.Product p JOIN Sales.SalesOrderDetail sod
        ON p.ProductID = sod.ProductID JOIN
        [Sales].[SalesOrderHeader] soh ON soh.SalesOrderID
        = sod.SalesOrderID WHERE p.Name LIKE 'Mountain-200
        Black, 38' AND soH.OrderDate >= '2013-01-01' AND
        soh.OrderDate < '2014-01-01'
        GROUP BY p.Name, MONTH(soh.[OrderDate]);
```

	Name	Quantity	Total Earnings	MonthNum
1	Mountain-200 Black, 38	93	15937,4814	1
2	Mountain-200 Black, 38	100	9222,7102	2
3	Mountain-200 Black, 38	141	12397,4883	3
4	Mountain-200 Black, 38	118	16194,0607	4
5	Mountain-200 Black, 38	86	26881,593	5
6	Mountain-200 Black, 38	181	47642,6897	6
7	Mountain-200 Black, 38	158	31938,4035	7
8	Mountain-200 Black, 38	89	36830,0183	8
9	Mountain-200 Black, 38	151	33679,6032	9
10	Mountain-200 Black, 38	134	42455,5898	10
11	Mountain-200 Black, 38	89	48763,9663	11
12	Mountain-200 Black, 38	130	53889,487	12

Besynnerligt är att färre sålda i månad 1, gett mer total inkomst än fler sålda i månad 2. Detta var jag tvungen att titta närmre på. Genom att söka i Sales.SalesOrderDetail efter ProductID (782), fann jag att UnitPrize var cirka 1229, förutom när CarrierTrackingNumber hade värdet NULL, då priset steg till ca 2049. Detta gäller t.ex. SalesOrderID 48410, SaleOrderDetailID 24185. När jag undersökt i SalesOrderHeader så kan jag se att de med högre pris har ett annat värde i kolumnen OnlineOrderFlag, samt en annan ShipMethod. Med andra ord så säljs de på annorlunda sätt, och detta borde undersökas närmre för att maximera de sätt som säljer bäst.



Medelvinsten per månad ligger mellan 21604.135113051223 och 41034.71345361546 med 95% konfidensgrad (se Python-skriptet). Det ser dock ut som att försäljningen/inkomsten bara ökar under årets gång, och när jag även tittade på förra året så styrker det en uppgång i försäljning. Detta gör att Konfidensintervallet kanske inte går att använda för att förutspå framtiden, men man skulle kunna följa hur många cyklar som säljs, så att produktion och lagerhållning är därefter.

Executive summary.

En analys har gjorts av företagets produkters försäljningssiffror, vad de har dragit in för summor, dess tillverkningskostnader, och de totala vinsterna som detta gett under det senaste fulla året (2013). Detta visade att ”Mountain-200 Black, 38” ger mest total vinst. Försäljningen för denna produkt verkar öka månadsvis, och det kan vara svårt att använda en medelvinst inom ett konfidensintervall för att förutspå framtida inkomster, men det bör går att istället följa kurvan och på så sätt se framåt, kanske med en medel-ökning. Det visar även att produkter som drar mycket pengar, inte nödvändigtvis är de som företaget tjänar mest på, tack vare tillverkningskostnaderna. I analysen upptäcktes flera produkter som har en negativ total vinst, detta tillsynes p.g.a. en högre produktionskostnad än vad produkterna säljer för. En rekommendation är att se över tillverkningskostnaderna för dessa produkterna, och/eller se över om de har något annat värde, t.ex. i reklamsyfte. Alternativ kan vara att höja priset, eller att helt sluta sälja vissa produkter/minska antalet som finns att sälja (då det kan se dåligt ut att helt ta bort en speciell storlek). Det kan finnas en möjlighet för företaget att spara pengar här, men det kan också vara viktigt att inte skapa missnöje genom att helt ta bort vissa produkter, så en undersökning om denna balansen föreslås.

Datum för muntlig presentation

Videolänk: https://youtu.be/xoKgPPHBT5k?si=EsLPv-T1PHrchd_n

Redogörelser

Utmaningar du haft under arbetet samt hur du hanterat dem.

Att få en överblick över databasen var svårt och tog tid tills jag löste det genom att göra ett diagram. Det gick snabbt att koppla Python till databasen tack vare videon. Att hitta något intressant att göra en analys av, tog lite tid också, men sedan valde jag att forsätta och utveckla något jag tittat på under min beskrivning av databasen. Det var roligt att jag fann något intressant som jag inte ens letade efter - produkterna som har för hög tillverkningskostnad.

Vilket betyg du anser att du skall ha och varför.

Jag är väldigt långt ifrån en expert, och det är säkert möjligt att göra super-avancerade saker med den här databasen, men jag känner ändå att jag kan analysera och diskutera till en viss grad, och jag är nöjd över min start i den här världen. Jag är inte rädd för att rota igenom en databas. Så någonstans mellan G och VG tror jag.

Tips du hade ”gett till dig själv” i början av kursen nu när du slutfört den.

Titta på fler olika databaser. Gör databas-diagram tidigare, även då krävs det nog mycket mer erfarenhet att få en bra överblick av en stor databas, men det hjälper i alla fall. Man känner sig lite halv-blind när man inte kan något om cyklar och dess delar, så man ska inte glömma att det på ett sätt kanske blir lättare om databasen är mer inom ens eget intresse/kunskapsområde.