**Database Management**

**&**

**Data Warehousing**

**Entity Relationship Model for Hospital Management:**

Diagram

Description automatically generated

Texas Tire & Battery Inc. is a wholesaler of automotive products. Company executives have recruited your team as a consultant to design and document a data warehouse to support decision making. Company has a chain of depots from which it fulfills customers’ orders. The data warehouse design should enable analysis of product sales in terms of dollar amounts and order quantities.

**Dimensional Model for Texas Tire & Battery Inc:**

Diagram

Description automatically generated

Diagram

Description automatically generated

**Demonstrates filters, sorting, aggregation, joins:**

1. List the *Store*, *City*, *State*, and *Zip code* for *stores located in California* and *Texas*. Display result-set in *ascending* order of states, and for each state in *ascending* order of city names. (**Note** – Please refer to the Store table).

SELECT DISTINCT STORE, CITY, STATE, ZIP\_CODE

FROM STORE

WHERE STATE IN ('CA','TX')

ORDER BY STATE ASC, CITY ASC;

1. List the *Customer ID (Cust\_ID)*, *Transaction amount (Tran\_Amt)*, *Transaction date (Tran\_Date)*, *City*, *State*, and *Zip code (Zip\_Code)* for the *largest purchase transaction (Tran\_Type)* by *customer from Texas (State)*. Information about the purchase transaction should be *displayed once*. (**Note** – Please refer to the Customer and Transact tables).

SELECT DISTINCT TOP 1 WITH TIES

CUSTOMER.CUST\_ID AS 'Customer ID',

TRAN\_AMT AS 'Transaction amount',

TRAN\_DATE AS 'Transaction date',

City,

State,

ZIP\_CODE AS 'Zip code'

FROM CUSTOMER

JOIN TRANSACT

ON CUSTOMER.CUST\_ID = TRANSACT.CUST\_ID

WHERE STATE = 'TX'

AND TRAN\_TYPE = 'P'

ORDER BY TRAN\_AMT DESC;

1. Display with appropriate column heading the *total purchase amount (Tran\_Type)* in *year 2015 (Tran\_Date)* at *stores located in Texas (State)*. (**Note** – Please refer to the Transact and Store tables).

SELECT STORE.STORE, SUM(TRAN\_AMT) AS 'TOTAL PURCHASE AMOUNT'

FROM STORE

JOIN TRANSACT

ON TRANSACT.STORE = STORE.STORE

WHERE STATE = 'TX'

AND YEAR(TRAN\_DATE) = '2015'

AND TRAN\_TYPE = 'P'

GROUP BY STORE.STORE

ORDER BY STORE.STORE;

1. For *stores located in Texas (State)*, what is the *total purchase amount (Tran\_Type)* for *each year (Tran\_Date)*. In the result-set, include two columns: *year* and *total dollar amount of purchase for each year*. (**Note** – Please refer to the Transact and Store tables).

SELECT

YEAR(TRAN\_DATE) AS 'Year',

SUM(TRAN\_AMT) AS 'Total dollar amount of purchase'

FROM STORE

JOIN TRANSACT

ON TRANSACT.STORE = STORE.STORE

WHERE STATE = 'TX'

AND TRAN\_TYPE = 'P'

GROUP BY YEAR(TRAN\_DATE)

ORDER BY Year ASC;

1. List the *Product SKU (SKU)*, *Department description (Dept\_Desc)*, *DeptDec\_Desc*, *Color*, *Retail price (Retail)*, and *Size (SKU\_Size) for navy (Color) career (DeptDec\_Desc) sweaters (Classification)* with *retail price (Retail) less than or equal to $100* from *Jones Signature* or *Daniel Cremieux (Dept\_Desc)*. Display result-set in *descending* order of *Department Description (Dept\_Desc)*, and for each Department Description in *ascending* order of *Size* (SKU\_SIZE). Information for each Product SKU should be *displayed once*. (**Note** – Please refer to the Department, SKU, and SKU\_Store tables).

SELECT DISTINCT

SKU.SKU AS 'Product SKU',

DEPT\_DESC AS 'Department description',

DeptDec\_Desc,

Color,

SKU\_STORE.RETAIL AS 'Retail price',

SKU\_SIZE AS 'Size'

FROM DEPARTMENT

JOIN SKU

ON DEPARTMENT.DEPT = SKU.DEPT

JOIN SKU\_STORE

ON SKU.SKU = SKU\_STORE.SKU

WHERE DEPT\_DESC IN ('Jones Signature','Daniel Cremieux')

AND COLOR LIKE '%NAVY%'

AND DEPTDEC\_DESC LIKE '%CAREER%'

AND CLASSIFICATION LIKE '%SWEATER%'

AND SKU\_STORE.RETAIL <=100

ORDER BY DEPT\_DESC DESC, SKU\_SIZE ASC;

1. In terms of the *Dept\_Desc*, which *perfume/cologne (Classification)* generated the *highest total dollar sales (Tran\_Amt)*. Display the *Dept\_Desc* and *total dollar sales amount*. (**Note** – Please refer to the Department, SKU, and Transact tables).

SELECT TOP 1 WITH TIES

Dept\_Desc,

SUM(TRAN\_AMT) AS 'Dollar sales amount'

FROM DEPARTMENT

JOIN SKU

ON DEPARTMENT.DEPT = SKU.DEPT

JOIN TRANSACT

ON SKU.ITEM\_ID = TRANSACT.ITEM\_ID

WHERE SKU.CLASSIFICATION = 'PERFUME/COLOGNE'

GROUP BY Dept\_Desc

ORDER BY SUM(TRAN\_AMT) DESC;

1. List the *Customer ID (Cust\_ID)*, *City*, *State*, and *Zip Code (Zip\_Code)* for customers from *Texas (State)* who have *total purchase of $15,000 or more (total of Tran\_Amt)*. Display these customers in *descending* order of *total purchase amount*. (**Note** – Please refer to the Customer and Transact tables).

SELECT DISTINCT

CUSTOMER.CUST\_ID AS 'Customer ID'

, City

, State

, ZIP\_CODE AS 'Zip code'

, SUM(TRAN\_AMT) AS TOTAL\_PURCHASE

FROM CUSTOMER

INNER JOIN TRANSACT

ON CUSTOMER.CUST\_ID = TRANSACT.CUST\_ID

WHERE STATE LIKE 'TX'

GROUP BY CUSTOMER.CUST\_ID, CITY, STATE, ZIP\_CODE

HAVING SUM(TRAN\_AMT) >= 15000

ORDER BY TOTAL\_PURCHASE DESC;

1. List the *Department Code (Dept)*, *Department Description (Dept\_Desc)*, *DeptDec\_Desc*, *DeptCent*, and *Deptcent\_Desc* for departments with *no items* in the *SKU* table. *Exclude* from the list, departments with *Lease* in the *Deptcent\_Desc* column. (**Note** – Please refer to the Department and SKU tables).

SELECT DEPARTMENT.DEPT AS 'Department Code',

DEPT\_DESC AS 'Department Description',

DeptDec\_Desc,

DeptCent,

Deptcent\_Desc

FROM DEPARTMENT

LEFT JOIN SKU

ON DEPARTMENT.DEPT = SKU.DEPT

WHERE SKU.DEPT IS NULL

AND DeptCent\_Desc <>'Lease';

1. For each store, list the *Store*, *City*, *State*, *Zip Code (Zip\_Code)*, and *number of items (SKU) with retail price (Retail) $500 or more*. Display result-set in *descending* order of the number of items with retail price $500 or more. (**Note** – Please refer to the Store and SKU\_Store tables).

SELECT DISTINCT

STORE.Store, City, State,

ZIP\_CODE AS 'Zip code',

COUNT(DISTINCT SKU) AS 'Number of Items'

FROM STORE

INNER JOIN SKU\_STORE

ON STORE.STORE = SKU\_STORE.STORE

WHERE RETAIL >=500

GROUP BY STORE.STORE, CITY, STATE, ZIP\_CODE

ORDER BY 'Number of Items' DESC;

**Demonstrates sub queries:**

1. List *CustomerID (Cust\_ID)*, *City*, *State*, *Zip code (Zip\_Code)*, and *Dept\_Desc* for customers from Texas who have purchased *both Chanel and Armani (Dept\_Desc) perfume/cologne (Classification).* Display each combination of customer and perfumes purchased *once*. Sort result-set in *ascending* order of city, and for each city in *ascending* order of zip code. (**Note** – Please refer to the Department, SKU, Transact, and Customer tables. For Dept\_Desc condition, use the LIKE operator with ‘%Chanel%’ and ‘%Armani%’.)

SELECT DISTINCT SELECTED\_CUSTOMERS.CUST\_ID, SELECTED\_CUSTOMERS.City,

SELECTED\_CUSTOMERS.State, SELECTED\_CUSTOMERS.ZIP\_CODE,

DEPARTMENT.DEPT\_DESC

FROM

(

SELECT DISTINCT CUSTOMER.CUST\_ID,

CUSTOMER.City, CUSTOMER.State, CUSTOMER.ZIP\_CODE

FROM CUSTOMER

WHERE CUSTOMER.CUST\_ID IN

(

SELECT DISTINCT CUSTOMER.CUST\_ID

FROM DEPARTMENT

JOIN SKU

ON DEPARTMENT.DEPT = SKU.DEPT

JOIN TRANSACT

ON SKU.SKU = TRANSACT.SKU

JOIN CUSTOMER

ON TRANSACT.CUST\_ID = CUSTOMER.CUST\_ID

WHERE (DEPT\_DESC LIKE '%Chanel%')

AND SKU.CLASSIFICATION = 'Perfume/Cologne'

AND STATE LIKE 'TX'

)

AND CUSTOMER.CUST\_ID IN

(

SELECT DISTINCT CUSTOMER.CUST\_ID

FROM DEPARTMENT

JOIN SKU

ON DEPARTMENT.DEPT = SKU.DEPT

JOIN TRANSACT

ON SKU.SKU = TRANSACT.SKU

JOIN CUSTOMER

ON TRANSACT.CUST\_ID = CUSTOMER.CUST\_ID

WHERE (DEPT\_DESC LIKE '%Armani%')

AND SKU.CLASSIFICATION = 'Perfume/Cologne'

AND STATE LIKE 'TX'

)

) SELECTED\_CUSTOMERS

LEFT JOIN TRANSACT

ON TRANSACT.CUST\_ID = SELECTED\_CUSTOMERS.CUST\_ID

LEFT JOIN SKU

ON TRANSACT.ITEM\_ID = SKU.ITEM\_ID

LEFT JOIN DEPARTMENT

ON DEPARTMENT.DEPT = SKU.DEPT

WHERE (DEPT\_DESC LIKE '%Chanel%' OR Dept\_Desc LIKE '%Armani%')

ORDER BY CITY ASC, Zip\_code ASC;

1. Write a SELECT statement to list the Account Number and Account Description for General Ledger Accounts that have no invoice line items posted yet. Display the AccountNo and AccountDescription in ascending order of AccountDescription. (Note – Please refer to the GLAccounts and InvoiceLineItems tables).

select AccountNo, AccountDescription

from GLAccounts

where GLAccounts.AccountNo not in (select InvoiceLineItems.AccountNo from InvoiceLineItems

where GLAccounts.AccountNo = InvoiceLineItems.AccountNo)

order by AccountDescription asc;

1. Write a SELECT statement to display InvoiceID, InvoiceDate, InvoiceTotal, TotalCredits (CreditTotal + PaymentTotal), TermsID, TermsDescription, and InvoiceLineItemAmount for invoices with TermsID 3 that are not fully paid (InvoiceTotal – PaymentTotal – CreditTotal > 0) and have InvoiceTotal greater than or equal to $500. Display the result-set in descending order of the InvoiceTotal values. (Note – Please refer to the Invoices, Terms, and InvoiceLineItems tables).

select Invoices.InvoiceID,InvoiceDate,InvoiceTotal, (PaymentTotal+CreditTotal) as TotalCredits, Terms.TermsID, TermsDescription, InvoiceLineItems.InvoiceLineItemAmount

from Invoices

inner join InvoiceLineItems on Invoices.InvoiceID = InvoiceLineItems.InvoiceID

inner join Terms on Invoices.TermsID = Terms.TermsID

where Terms.TermsID = 3

and (InvoiceTotal - PaymentTotal - CreditTotal)>0

and InvoiceTotal >= 500

order by InvoiceTotal desc;

1. Display the VendorID, VendorName, InvoiceID, InvoiceDate and InvoiceTotal for invoice(s) with invoice total amount greater than the largest invoice total amount for invoices from vendors located in California. Display the result-set in descending order of InvoiceTotal values. (Note – Please refer to the Vendors and Invoices tables).

select Vendors.VendorID, VendorName, InvoiceID, InvoiceDate , InvoiceTotal from

Vendors inner join Invoices on Vendors.VendorID = Invoices.VendorID

where Invoices.InvoiceTotal > (select Max(Invoices.InvoiceTotal) from

Vendors inner join Invoices on Vendors.VendorID = Invoices.VendorID AND Vendors.VendorState = 'CA')

order by InvoiceTotal desc;

1. Write a SELECT statement to display the TermsID, TermsDescription, Total of Invoice Amounts (i.e., total of invoice total values), Number of Invoices, and Average Invoice Total for each payment term that has ten or more invoices. Display the result-set in ascending order of Average Invoice Total values. In the result-set, provide appropriate column names for the total of invoice amounts, number of invoices, and average invoice total columns. (Note – Please refer to the Terms and Invoices tables).

select Terms.TermsID, TermsDescription,sum(InvoiceTotal) as 'Total of Invoice Amounts', count (\*) as 'NumberOfInvoices', avg(InvoiceTotal) as 'AverageInvoiceTotal'

from Invoices inner join Terms on Invoices.TermsID = Terms.TermsID

group by Terms.TermsID,TermsDescription

having count(\*) >= 10

order by 'AverageInvoiceTotal';

1. Write a SELECT statement to display the VendorID and VendorName for vendors with invoices in February 2012 and March 2012. Display the result-set in ascending order of VendorName values. In the result-set each vendor should be displayed once. (Note – Please refer to the Vendors and Invoices tables).

select distinct Vendors.VendorID, Vendors.VendorName

from Vendors inner join Invoices on Vendors.VendorId = Invoices.VendorID

where (year (InvoiceDate) = '2012' and month(InvoiceDate) = '02')

or (year (InvoiceDate) = '2012' and month(InvoiceDate) = '03')

And VENDORs.vendorId in (select distinct vendors.vendorID

from Vendors inner join Invoices on Vendors.VendorId = Invoices.VendorID

where (year (InvoiceDate) = '2012' and month(InvoiceDate) = '02')

And

VENDORs.vendorId in (select distinct vendors.vendorID

from Vendors inner join Invoices on Vendors.VendorId = Invoices.VendorID

where (year (InvoiceDate) = '2012' and month(InvoiceDate) = '03')))

order by VendorName asc;