

SELCT QUREY

Exercise_DML_SelectWhere

Tryout 1: Exercise_DML_SelectWhere

Problem Statement

Write a query to display product id, brand, product category and color of all clothing type products.

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT productid, brand, pcategory, color  
FROM Product WHERE pcategory = 'Clothing';
```

```
HSQl +  
1: SELECT productid, brand, pcategory, color  
2: FROM Product WHERE pcategory = 'Clothing';
```

Exercise_DML_RatingNotAvailable

Tryout 1: Exercise_DML_RatingNotAvailable

Problem Statement

Write a query to display vendorid, vendorname and rating of those vendors that do not have any rating.

```
SELECT Vendorid, Vendorname, Rating FROM Vendor WHERE Rating is NULL
```

```
HSQl +  
1: SELECT Vendorid, Vendorname, Rating FROM Vendor WHERE Rating is NULL
```

Exercise_DML_UniqueCategory

Tryout 1: Exercise_DML_UniqueCategory

Problem Statement

Write a query to display the categories of all the products.
Remove the duplicate records from the output if any.

```
SELECT DISTINCT Pcategory FROM Product;
```

```
HSQl +  
1: SELECT DISTINCT Pcategory FROM Product;
```

```
SELECT Id, EName FROM Employee WHERE EName LIKE 'E%';
```

```
SELECT Id, EName, DOJ FROM Employee WHERE DOJ LIKE '___-__-__';
```

Exercise_DML_LIKE

Tryout 1: Exercise_DML_LIKE

Problem Statement

Write a query to display CustId, CustName, Age and Email of those customers who have 'a' in the CustName at any position and 'u' in the CustLocation at any position.

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT custid,custname, age, email  
FROM Customer  
WHERE custname LIKE '%a%' AND custlocation LIKE '%u%'
```

```
HSQl +  
1: SELECT custid,custname, age, email  
2: FROM Customer  
3: WHERE custname LIKE '%a%' AND custlocation LIKE '%u%'
```

Assignment_DML_DisplayJournals

Summary Questions

Tryout 1: Assignment_DML_DisplayJournals

Problem Statement

Identify the journals from 'Scientific' or 'Politics' genre. Display the journalid and journalname of those journals.

The following records will appear as a part of output along with other record(s).

Sample Output:

```
SELECT journalid, journalname
FROM Journal
WHERE Genre IN ('Scientific','Politics');
```

Assignment_DML_DisplayDiscountedAmount

Summary Questions

Tryout 1: Assignment_DML_DisplayDiscountedAmount

Problem Statement

The publishers want to give a discount of 5% on monthly charges. Write a query to display the following:

1. journal name
2. current monthly charges
3. monthly charges applicable after discount
4. discount in INR

Hint: Use the formula MonthlyCharges*(1-5/100) and MonthlyCharges*(5/100) to find monthly charges applicable after discount and discount in INR.

```
HSQl
1 SELECT journalid, journalname
2 FROM Journal
3 WHERE Genre IN ('Scientific','Politics');
```

```
HSQl
1 SELECT journalname,
2 monthlycharges AS "CURRENT CHARGES",
3 (monthlycharges * (1-5/100)) AS "OFFER PRICE",
4 (monthlycharges * (5/100)) AS "DISCOUNTED AMOUNT"
5 FROM Journal
```

Result

```
SELECT journalname,
monthlycharges AS "CURRENT CHARGES",
(monthlycharges * (1-5/100)) AS "OFFER PRICE",
(monthlycharges * (5/100)) AS "DISCOUNTED AMOUNT"
FROM Journal
```

Assignment_DML_DatesRange

Summary Questions

Tryout 1: Assignment_DML_DatesRange

Problem Statement

Retrieve subscriptionid and customerid of the customers who have their startdate between the dates '1-Jan-2023' and '30-Jun-2023'.

The following records will appear as a part of output along with other record(s).

Sample Output:

```
HSQl
1 SELECT subscriptionid, customerid
2 FROM Subscription
3 WHERE startdate BETWEEN '2023-01-01' AND '2023-06-30';
```

```
SELECT Subscriptionid, customerid
FROM Subscription
WHERE startdate BETWEEN '2023-01-01' AND '2023-06-30';
```

Assignment_DML_DisplayNonRegularCustomers

Summary Questions

Tryout 1: Assignment_DML_DisplayNonRegularCustomers

Problem Statement

Display customerid, customername, state and city of customers whose type is other than Regular.

The following records will appear as a part of output along with other record(s).

Sample Output:

```
HSQl
1 SELECT Customerid, customername, state, city
2 FROM Customer
3 WHERE customertype <> 'Regular';
```

```
SELECT Customerid, customername, state, city
FROM Customer
```

```
WHERE customertype <> 'Regular';
```

Assignment_DML_SQLOperators

Summary Questions

Tryout 1: Assignment_DML_SQLOperators

Problem Statement

Retrieve the journal name and genre of journals whose genre is either 'Trade' or 'Politics' and monthly charges lie in the range of 90 to 125 (both inclusive).

For the given sample data, the following record will feature as part of the output along with other record(s).

Sample Output:

```
SELECT journalname, genre
FROM Journal
WHERE Genre IN ('Trade','Politics') AND monthlycharges BETWEEN 90 AND 125;
```

Assignment_DML_Operators

Summary Questions

Tryout 1: Assignment_DML_Operators

Problem Statement

Display CustomerId, CustomerName, CustomerType and State of the customer(s) from Illinois who have 'prime' or 'gulf' in their customertype.

The following records will appear as a part of output along with other record(s).

Sample Output:

```
SELECT customerid, customername, customertype, state
FROM Customer
WHERE state = 'Illinois' AND (customertype LIKE '%prime%' OR customertype LIKE '%gulf');
```

Assignment_DML_MissingContactNos

Summary Questions

Tryout 1: Assignment_DML_MissingContactNos

Problem Statement

Retrieve the publishername of publishers who do not have contact number.

The following records will appear as a part of output:

```
SELECT Publishername
FROM Publisher
WHERE contactno IS NULL
```

Assignment_DML_MonthlyChargesRange

Summary Questions

Tryout 1: Assignment_DML_MonthlyChargesRange

Problem Statement

Retrieve journal id, journal name and monthly charges for journals that have monthly charges in the range 85 to 130 (excluding these values).

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT Journalid, journalname, monthlycharges
FROM Journal
WHERE monthlycharges BETWEEN 86 AND 129;
```

Assignment_DML_SelectiveGenre

Summary Questions < >

Tryout 1: Assignment_DML_SelectiveGenre

Problem Statement

Retrieve the journalid and monthlycharges of all 'Trade' and 'Politics' genre.

The following records will appear as a part of output along with other record(s).

Sample Output:

```
SELECT Journalid, monthlycharges  
FROM Journal  
WHERE Genre IN ('Trade', 'Politics');
```

HSQL

```
1 SELECT Journalid, monthlycharges  
2 FROM Journal  
3 WHERE Genre IN ('Trade', 'Politics');
```

A- A+ C

Assignment_DML_UniqueStates

Summary Questions < >

Tryout 1: Assignment_DML_UniqueStates

Problem Statement

Display the names of states in which the customers reside.

Note: Display unique names.

The following records will appear as a part of output along with other record(s).

Sample Output:

```
SELECT DISTINCT state  
FROM Customer;
```

HSQL

```
1 SELECT DISTINCT state  
2 FROM Customer;
```

A- A+ C

Assignment_DML_UniqueStatePerCustomerType

Summary Questions < >

Tryout 1: Assignment_DML_UniqueStatePerCustomerType

Problem Statement

For each customer type display the states in which customers reside.

Note: Display unique values.

```
SELECT DISTINCT Customertype, state  
FROM Customer;
```

HSQL

```
1 SELECT DISTINCT Customertype, state  
2 FROM Customer;
```

A- A+ C

Assignment_DML_Like_ti

Summary Questions < >

Tryout 1: Assignment_DML_Like_ti

Problem Statement

Display JournalName and Genre of journal(s) whose Genre contains the string 'ti'.

The following record will appear as a part of output along with other record(s).

Sample Output:

```
SELECT journalname, genre  
FROM Journal  
WHERE Genre LIKE '%ti%';
```

HSQL

```
1 SELECT journalname, genre  
2 FROM Journal  
3 WHERE Genre LIKE '%ti%';
```

A- A+ C

Assignment_DML_LIKE

Summary Questions < >

Tryout 1: Assignment_DML_LIKE

Problem Statement

Retrieve publisher details for publishers whose names have 'Y' as the second last character.

The following records will appear as a part of output along with other record(s).
Sample Output:

```
SELECT publisherid, publishername, rating, contactno
FROM Publisher
WHERE publishername LIKE '%i_Y';
```

HSQL

```
1: SELECT publisherid, publishername, rating, contactno
2: FROM Publisher
3: WHERE publishername LIKE '%i_Y';
```

Exercise_Functions_CeilingFloor

Summary Questions < >

Tryout 1: Exercise_Functions_CeilingFloor

Problem Statement

For every product offered by the vendor, display the VendorId as "VendorId", price values rounded off to the nearest higher and lower whole numbers (integers) as "Higher_Round" and "Lower_Round" for every product-vendor combination.

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT vendorid AS "VendorId",
CEIL(price) AS "Higher_Round",
FLOOR(price) AS "Lower_Round"
FROM VendorOffering
```

HSQL

```
1: SELECT vendorid AS "VendorId",
2: CEIL(price) AS "Higher_Round",
3: FLOOR(price) AS "Lower_Round"
4: FROM VendorOffering;
```

Exercise_Functions_Absolute

Summary Questions < >

Tryout 1: Exercise_Functions_Absolute

Problem Statement

Display the price of each product offered by the vendor after applying a discount of 500 rupees as "After Discount". The price should be displayed as positive values.

```
SELECT ABS(price-500) AS "After Discount"
FROM VendorOffering
```

HSQL

```
1: SELECT ABS(price-500) AS "After Discount"
2: FROM VendorOffering
```

Exercise_Functions_FilterProducts

Summary Questions < >

Tryout 1: Exercise_Functions_FilterProducts

Problem Statement

Identify all white colored products whose product name has more than 9 characters. Display numeric part of ProductId and PName separated by hyphen (-) for the identified products. (Do case-insensitive comparison).

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT (productId || '-' || Pname) AS "Product"
FROM Product
WHERE LOWER(color) = 'white' AND LENGTH(Pname) > 9;
```

HSQL

```
1: SELECT (productId || '-' || Pname) AS "Product"
2: FROM Product
3: WHERE LOWER(color) = 'white' AND LENGTH(Pname) > 9;
```

Exercise_Functions_Character

Summary Questions < >

Tryout 1: Exercise_Functions_Character

Problem Statement

Display the product name along with the first character of its review separated by hyphen (-) for those products whose category contains the character 'I' anywhere. Display it as "Product_Review".

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT (Pname || '-' || SUBSTR(review,1,1)) AS "Product_Review"  
FROM Product  
WHERE pcategory LIKE '%i%';
```

SQL +

```
1: SELECT (Pname || '-' || SUBSTR(review,1,1)) AS "Product_Review"  
2: FROM Product  
3: WHERE pcategory LIKE '%i%';
```

A- A+ C G

Result

Exercise_Functions_Aggregate

Summary Questions < >

Tryout 1: Exercise_Functions_Aggregate

Problem Statement

Identify the vendors whose rating is known. For the identified vendors, display the minimum rating as "MIN_RATING", maximum rating as "MAX_RATING", average rating as "AVG_RATING" and number of vendors as "NUMBEROFVENDORS".

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT MIN(rating) AS min_rating,  
MAX(rating) AS max_rating,  
AVG(Rating) AS avg_rating,  
COUNT(rating) AS numberofvendors  
FROM vendor
```

SQL +

```
1: SELECT MIN(rating) AS min_rating,  
2: MAX(rating) AS max_rating,  
3: AVG(Rating) AS avg_rating,  
4: COUNT(rating) AS numberofvendors  
5: FROM vendor
```

A- A+ C G

Exercise_Functions_TotalPrice

Summary Questions < >

Tryout 1: Exercise_Functions_TotalPrice

Problem Statement

Round off the prices of all the products to the next higher integer and display their total rounded off as "Total_Price".

```
SELECT SUM(CEIL(price)) AS "Total_Price" FROM Vendoroffering;
```

SQL +

```
1: SELECT SUM(CEIL(price)) AS "Total_Price" FROM Vendoroffering;
```

A- A+ C G

Exercise_Functions_MonthsLapsed

Summary Questions < >

Tryout 1: Exercise_Functions_MonthsLapsed

Problem Statement

Display invoice id, product id and months lapsed between current date and purchase date for the product 'P105'.

For the given sample data, the following is the expected output observed on 2024-08-27. Actual output might vary on the basis of current system date.

```
SELECT invoiceid, productid, ROUND(MONTHS_BETWEEN(CURRENT_DATE,Purchasedate)) AS  
noofmonths  
FROM Invoice  
WHERE productid = 'P105';
```

SQL +

```
1: SELECT invoiceid, productid, ROUND(MONTHS_BETWEEN(CURRENT_DATE,Purchasedate)) AS noofmonths  
2: FROM Invoice  
3: WHERE productid = 'P105';
```

A- A+ C G

Exercise_Functions_Date

Summary Questions < >

Tryout 1: Exercise_Functions_Date

Problem Statement

Assume that in the supermarket scenario, every product has an exchange validity of 2 months, starting from the date of purchase.
Display the InvoiceId as "InvoiceId", PurchaseDate as "PurchaseDate" and exchange validity date as "ExchangeValidityDate" of all the sold products.

```
SELECT InvoiceId AS "InvoiceId", PurchaseDate AS "PurchaseDate",
       ADD_MONTHS(PurchaseDate, 2) AS "ExchangeValidityDate"
  FROM Invoice
```

SQL +

```
1: SELECT InvoiceId AS "InvoiceId", PurchaseDate AS "PurchaseDate",
2:      ADD_MONTHS(PurchaseDate, 2) AS "ExchangeValidityDate"
3:  FROM Invoice
```

A- A+ C G

Exercise_Function_NVL

Summary Questions < >

Tryout 1: Exercise_Function_NVL

Problem Statement

Display the CustId as "CustomerId", CustName as "CustomerName", Email as "CustEmail" and Age as "CustAge" of all customers. If the email and age are unknown, display "Not Provided" and 0 respectively.

```
SELECT CustId AS "CustomerId", CustName AS "CustomerName",
       NVL>Email, 'Not Provided') AS "CustEmail",
       NVL(Age, 0) AS "CustAge"
  FROM Customer
```

SQL +

```
1: SELECT CustId AS "CustomerId", CustName AS "CustomerName",
2:        NVL>Email, 'Not Provided') AS "CustEmail",
3:        NVL(Age, 0) AS "CustAge"
4:   FROM Customer
```

A- A+ C G

Exercise_Functions_StringInCharacterColumn

Summary Questions < >

Tryout 1: Exercise_Functions_StringInCharacterColumn

Problem Statement

Display VendorId and Rating as "RATING" of all the vendors. If the rating is unknown, display "Not Provided".

For the given sample data, the following records will feature as part of the output along with other records).

Sample Output:

```
SELECT vendorId, NVL(TO_CHAR(Rating), 'Not Provided') AS RATING
  FROM Vendor
```

SQL +

```
1: SELECT vendorId, NVL(TO_CHAR(Rating), 'Not Provided') AS RATING
2:   FROM Vendor
```

A- A+ C

Assignment_Functions_Numeric

Summary Questions < >

Tryout 1: Assignment_Functions_Numeric

Problem Statement

Apply a discount of 70.25 rupees on the MonthlyCharges for those journals whose Genre contains more than 8 characters. Display the JournalId as "JournalId" and round off the calculated price to the lower(integer) and display as "Rounded_Integer".

```
SELECT JournalId AS "JournalId", FLOOR(monthlycharges - 70.25) AS "Rounded_Integer"
  FROM Journal
 WHERE LENGTH(Genre) >8;
```

SQL +

```
1: SELECT JournalId AS "JournalId", FLOOR(monthlycharges - 70.25) AS "Rounded_Integer"
2:   FROM Journal
3: WHERE LENGTH(Genre) >8
```

A- A+ C

Assignment_Functions_CountSelectivePublishers

A E G

Summary Questions < >

Tryout 1: Assignment_Functions_CountSelectivePublishers

Problem Statement

Identify the publishers whose contact number is known and their name contains more than 5 characters. Display the number of such publishers as "NUMBEROFPUBLISHERS".

For the given sample data, the following is the expected output.
Sample Output:

```
SELECT COUNT(*) AS NUMBEROFPUBLISHERS
FROM Publisher
WHERE Contactno IS NOT NULL AND LENGTH(publishername) > 5;
```

HSQL +

```
1: SELECT COUNT(*) AS NUMBEROFPUBLISHERS
2: FROM Publisher
3: WHERE Contactno IS NOT NULL AND LENGTH(publishername) > 5;
```

A- A+ G G

Assignment_Functions_RoundOff2DecimalDigits

A E G

Summary Questions < >

Tryout 1: Assignment_Functions_RoundOff2DecimalDigits

Problem Statement

The monthly charges of 'Academics' Journals is reduced by 37.5%. Display the JournalId as "JournalId", Genre as "Genre", existing monthlycharges as "Old_Price", and revised monthlycharges as "New_Price" for all the 'Academic' Journals. Revised Monthly charges should be rounded up to 2 decimal places.

Note: To calculate New_Price use (MonthlyCharges*0.625)

For the given sample data, the following is the expected output.

```
SELECT JournalID AS "JournalId",
Genre AS "Genre",
monthlycharges AS "Old_Price",
ROUND(monthlycharges*0.625, 2) AS "New_Price"
FROM Journal
WHERE genre = 'Academics';
```

HSQL +

```
1: SELECT JournalID AS "JournalId",
2: Genre AS "Genre",
3: monthlycharges AS "Old_Price",
4: ROUND(monthlycharges*0.625, 2) AS "New_Price"
5: FROM Journal
6: WHERE genre = 'Academics';
```

Result

Assignment_Functions_regularCustomers

A E G

Summary Questions < >

Tryout 1: Assignment_Functions_regularCustomers

Problem Statement

Display CustomerId and CustomerName of 'Regular' customer(s) who reside in a state that contains character 'o' anywhere.
(Do case-insensitive comparison for CustomerType)

For the given sample data, the following records will feature as part of the output along with other record(s).

```
SELECT customerid, customername
FROM Customer
WHERE LOWER(customertype) = 'regular' AND State LIKE '%o%';
```

HSQL +

```
1: SELECT customerid, customername
2: FROM Customer
3: WHERE LOWER(customertype) = 'regular' AND State LIKE '%o%';
```

Assignment_Functions_SelectiveJournals

Summary Questions

Tryout 1: Assignment_Functions_SelectiveJournals

Problem Statement

Display JournalId and JournalName for all the journal(s) whose last digit of the publisher id is more than 2 and journal name contains character 'o' and 't' anywhere, but in the same order.

For the given sample data, the following records will feature as part of the output along with other record(s).

```
SELECT JournalId, journalname
FROM Journal
WHERE SUBSTR(publisherid, LENGTH(publisherid)) > 2 AND journalname LIKE '%o%t%';
```

HSQL

```
1 SELECT JournalId, journalname
2 FROM Journal
3 WHERE SUBSTR(publisherid, LENGTH(publisherid)) > 2 AND journalname LIKE '%o%t%';
```

A- A+

Assignment_Functions_Aggregate

Summary Questions

Tryout 1: Assignment_Functions_Aggregate

Problem Statement

Display Maximum, Minimum and Average values of DiscountPercent which is offered on any subscription. The display should have the column names as "MaxDiscount", "MinDiscount" and "AvgDiscount".

For the given sample data, the following is the expected output.

```
SELECT MAX(Discountpercent) AS "MaxDiscount",
       MIN(Discountpercent) AS "MinDiscount",
       AVG(Discountpercent) AS "AvgDiscount"
  FROM Subscription;
```

HSQL

```
1 SELECT MAX(Discountpercent) AS "MaxDiscount",
2      MIN(Discountpercent) AS "MinDiscount",
3      AVG(Discountpercent) AS "AvgDiscount"
4  FROM Subscription;
```

Assignment_Functions_CombineColumnData

Summary Questions

Tryout 1: Assignment_Functions_CombineColumnData

Problem Statement

Display CustomerId along with the first four character of their CustomerType separated by slash (/) as "Customer-ID".

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT Customerid || '/' || SUBSTR(customertype,1,4) AS "Customer-ID"
  FROM Customer;
```

HSQL

```
1 SELECT Customerid || '/' || SUBSTR(customertype,1,4) AS "Customer-ID"
2  FROM Customer
```

Assignment_Functions_CountUniqueStartDates

Summary Questions

Tryout 1: Assignment_Functions_CountUniqueStartDates

Problem Statement

Display the total number of unique StartDates as "Total_Dates" on which the subscription is taken.

For the given requirements, display UNIQUE records wherever applicable.

For the given sample data, the following is the expected output.

```
SELECT COUNT(DISTINCT StartDate) AS "Total_Dates"
  FROM Subscription;
```

HSQL

```
1 SELECT COUNT(DISTINCT StartDate) AS "Total_Dates"
2  FROM Subscription;
```

Assignment_Functions_MinDiscountPercent

Summary Questions

Tryout 1: Assignment_Functions_MinDiscountPercent

Problem Statement

For the Subscriptions bought in the month of November, display the minimum discount(percent) offered as "Min Discount Percent", along with the total number of customers who subscribed in that month as "No. Of Customers".

```
SELECT MIN(Discountpercent) AS "Min Discount Percent",
       COUNT(*) AS "No. Of Customers"
  FROM Subscription
 WHERE MONTH(Startdate) = 11;
```

HSQL

```
1 SELECT MIN(Discountpercent) AS "Min Discount Percent",
       COUNT(*) AS "No. Of customers"
  FROM Subscription
 WHERE MONTH(Startdate) = 11;
```

Assignment_Functions_NoOfUniqueCustomers

Summary Questions

Tryout 1: Assignment_Functions_NoOfUniqueCustomers

Problem Statement

Find the number of unique customers who have subscribed to a journal in any month except 'June'. Display the identified number of unique customers as "NoOfUniqueCustomer".

For the given requirements, display UNIQUE records wherever applicable.

For the given sample data, the following is the expected output.

HSQL

```
1 SELECT COUNT(DISTINCT customerId) AS "NoOfUniqueCustomer"
 2 FROM Subscription
 3 WHERE MONTH(Startdate) <> 06;
```

Result

```
SELECT COUNT(DISTINCT customerId) AS "NoOfUniqueCustomer"
  FROM Subscription
 WHERE MONTH(Startdate) <> 06;
```

Assignment_Functions_TotalMonthlyCharges

Summary Questions

Tryout 1: Assignment_Functions_TotalMonthlyCharges

Problem Statement

Display the total MonthlyCharges as "Monthly_Charges" of all the journals whose last digit of the numeric part of PublisherId is more than 4.

Note: The size of column PublisherId is as per the given table structure.

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT SUM(monthlyCharges) AS "Monthly_Charges"
  FROM Journal
 WHERE SUBSTR(PublisherId, LENGTH(PublisherId)) >4;
```

HSQL

```
1 SELECT SUM(monthlyCharges) AS "Monthly_Charges"
 2 FROM Journal
 3 WHERE SUBSTR(PublisherId, LENGTH(PublisherId)) >4;
```

Result

Assignment_Functions_NestedDateFunctions

Summary Questions

Tryout 1: Assignment_Functions_NestedDateFunctions

Problem Statement

On 30th June 2024, an executive is asked to send subscription renewal reminder to the customers whose subscription is going to end within a month. For this, he wants to identify the subscription id and corresponding customer id.

Help him write this query.

For the given sample data, the following is the expected output.

```
SELECT subscriptionID, Customerid
  FROM Subscription
```

HSQL

```
1 SELECT subscriptionID, Customerid
 2 FROM Subscription
 3 WHERE ADD_MONTHS(startdate,durationinmonths) BETWEEN '2024-06-30' AND '2024-07-30'
```

```
WHERE ADD_MONTHS(Startdate,durationinmonths) BETWEEN '2024-06-30' AND '2024-07-30';
```

Assignment_Case_JournalCategory

Summary Questions < >

Tryout > 1: Assignment_Case_JournalCategory

Problem Statement

A publisher wants to categorize the journals based on the monthly charges of the journals. Write a query to display JournalName, Monthly Charges and JournalCategory. The categorization is done as: If the MonthlyCharges is less than 75, then the JournalCategory will be 'LowCost'. Else if, MonthlyCharges is between 75 and 100, then the JournalCategory will be 'AverageCost'. Else, the JournalCategory will be 'HighCost'.

```
SELECT Journalname, monthlycharges,
(CASE WHEN monthlycharges<75 THEN 'LowCost'
WHEN monthlycharges<100 THEN 'AverageCost'
ELSE 'HighCost'
END)AS journalcategory
FROM Journal;
```

```
HSQL ▾
1 SELECT Journalname, monthlycharges,
2   (CASE WHEN monthlycharges<75 THEN 'LowCost'
3     WHEN monthlycharges<100 THEN 'AverageCost'
4     ELSE 'HighCost'
5   END)AS journalcategory
6 FROM Journal
```

ORDER BY

Exercise_Sorting_PCategory_Review

Summary Questions < >

Tryout > 1: Exercise_Sorting_PCategory_Review

Problem Statement

Display product id, product category and brand of the products which belongs to the category of 'Clothing' or 'Accessories' in ascending order of the product category and review.

Note: Use the concept of positional sorting wherever applicable.

```
SELECT productid, Pcategory, brand
FROM product
WHERE Pcategory IN ('Clothing', 'Accessories')
ORDER BY Pcategory, review;
```

```
HSQL ▾
1 SELECT productid, Pcategory, brand
2 FROM product
3 WHERE Pcategory IN ('Clothing', 'Accessories')
4 ORDER BY Pcategory, review;
```

Exercise_Sorting_PCategory_Color

Summary Questions < >

Tryout > 1: Exercise_Sorting_PCategory_Color

Problem Statement

Display the product id, product category, brand and color of all products in descending order of product category and ascending order of color.

```
SELECT productid, pccategory, brand, color
FROM Product
ORDER BY pccategory DESC,color;
```

```
HSQL ▾
1 SELECT productid, pccategory, brand, color
2 FROM Product
3 ORDER BY pccategory DESC,color;
```

Assignment_Sorting_Rating_Name

Summary Questions < >

Tryout 1: Assignment_Sorting_Rating_Name

Problem Statement

Display PublisherId, PublisherName and Rating of all the publishers in ascending order of Rating and PublisherName.

Note: Implement the concept of positional sorting wherever applicable.

HSQL ▾

```
1 SELECT publisherid, publishername, rating
2 FROM Publisher
3 ORDER BY rating, publishername;
```

```
SELECT publisherid, publishername, rating
FROM Publisher
ORDER BY rating, publishername;
```

Assignment_Sorting_Duration_StartDate

Summary Questions < >

Tryout 1: Assignment_Sorting_Duration_StartDate

A- A+ C

Problem Statement

Display the SubscriptionId, DurationInMonths and DiscountPercent of the Subscription(s) for which the DurationInMonths is more than 6 and arrange them in increasing order of DurationInMonths and decreasing order of StartDate.

HSQL ▾

```
1 SELECT subscriptionid,durationinmonths,discountpercent
2 FROM Subscription
3 WHERE durationinmonths > 6
4 ORDER BY durationinmonths,startdate DESC;
```

```
SELECT subscriptionid,durationinmonths,discountpercent
FROM Subscription
WHERE durationinmonths > 6
ORDER BY durationinmonths,startdate DESC;
```

GROUP BY HAVING

Exercise_GroupBy_LocationWiseMinAge

Summary Questions < >

Tryout 1: Exercise_GroupBy_LocationWiseMinAge

Problem Statement

Find the minimum age of customers in each Location.

For the given sample data, the following rows will appear as a part of output along with other rows.

HSQL ▾

```
1 SELECT custlocation AS "LOCATION", MIN(Age) AS minimumage
2 FROM Customer
3 GROUP BY custlocation;
```

```
SELECT custLocation AS "LOCATION", MIN(Age) AS minimumage
FROM Customer
GROUP BY custLocation;
```

Exercise_GroupBy_RecentPurchaseDate

Summary Questions < >

Tryout 1: Exercise_GroupBy_RecentPurchaseDate

Problem Statement

Find the recent date on which a customer has purchased any product.

For the given sample data, the following rows will appear as a part of output along with other records.

HSQL ▾

```
1 SELECT custID, MAX(purchasedate) AS recentpurchasedate
2 FROM Invoice
3 GROUP BY custID;
```

```
SELECT custID, MAX(purchasedate) AS recentpurchasedate
FROM Invoice
```

```
GROUP BY custID;
```

Exercise_GroupBy_WhiteProducts

Summary

Questions

Tryout 1: Exercise_GroupBy_WhiteProducts

Problem Statement

Find the number of White colored products in each brand.

For the given sample data, the following is the expected output.

Expected Output:

HSQL ▾

```
1 SELECT brand, COUNT(*) AS noofwhiteproducts
2 FROM Product
3 WHERE Color = 'White'
4 GROUP BY brand;
```

```
SELECT brand, COUNT(*) AS noofwhiteproducts
FROM Product
WHERE Color = 'White'
GROUP BY brand;
```

Exercise_GroupBy_RecentPurchaseDateProductWise

Summary

Questions

Tryout 1: Exercise_GroupBy_RecentPurchaseDateProductWise

Problem Statement

Find the most recent purchase date for the products purchased by customers.

For the given sample data, the following rows will appear as a part of output along with other records.

Sample Output:

HSQL ▾

```
1 SELECT custId, Productid, MAX(purchasedate) AS recentpurchasedate
2 FROM Invoice
3 GROUP BY custId,Productid;
```

Result

```
SELECT custId, Productid, MAX(purchasedate) AS recentpurchasedate
FROM Invoice
GROUP BY custId,Productid;
```

Exercise_GroupBy_LocationWiseMinAgeOfEachGender

Summary

Questions

Tryout 1: Exercise_GroupBy_LocationWiseMinAgeOfEachGender

Problem Statement

For each location, find the minimum age of each gender. Arrange the data in ascending order of location and gender.

For the given sample data, the following is the expected output.

Expected Output:

HSQL ▾

```
1 SELECT custlocation AS "LOCATION", Gender, MIN(AGE) AS minimumage
2 FROM Customer
3 GROUP BY custlocation,Gender
4 ORDER BY custlocation,Gender;
```

```
SELECT custlocation AS "LOCATION", Gender, MIN(AGE) AS minimumage
FROM Customer
GROUP BY custlocation,Gender
ORDER BY custlocation,Gender;
```

Exercise_GroupBy_MinAgeAtleast30

Summary Questions < >

Tryout 1: Exercise_GroupBy_MinAgeAtleast30

Problem Statement

Find the location(s) where minimum age of customer is atleast 30.

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT custlocation AS "LOCATION", MIN(age) AS minimumage
FROM Customer
GROUP BY custlocation
HAVING MIN(age) >=30;
```

```
HSQl
1 SELECT custlocation AS "LOCATION", MIN(age) AS minimumage
2 FROM Customer
3 GROUP BY custlocation
4 HAVING MIN(age) >=30;
```

Exercise_GroupBy_LoyalCustomers

Summary Questions < >

Tryout 1: Exercise_GroupBy_LoyalCustomers

Problem Statement

Identify the loyal customers.

Loyal customers are those who have purchased same product from same vendor more than once.

```
HSQl
1 SELECT custid,vendorid,productid, COUNT(*) AS nofpurchases
2 FROM Invoice
3 GROUP BY custid,vendorid,productid
4 HAVING COUNT(*) > 1;
```

```
SELECT custid,vendorid,productid, COUNT(*) AS nofpurchases
FROM Invoice
GROUP BY custid,vendorid,productid
HAVING COUNT(*) > 1;
```

Exercise_GroupBy_Month

Summary Questions < >

Tryout 1: Exercise_GroupBy_Month

Problem Statement

Display abbreviated name of month of purchase as "MONTH" (column alias) and number of purchases done in that month as "NUMBER_OF_PURCHASES" (column alias). Sort the data in descending order of number of purchases.

```
HSQl
1 SELECT TO_CHAR(Purchasedate,'MON') AS "MONTH", COUNT(*) AS "NUMBER_OF_PURCHASES"
2 FROM Invoice
3 GROUP BY TO_CHAR(Purchasedate,'MON')
4 ORDER BY NUMBER_OF_PURCHASES DESC;
```

```
SELECT TO_CHAR(Purchasedate, 'MON') AS "MONTH", COUNT(*) AS "NUMBER_OF_PURCHASES"
FROM Invoice
GROUP BY TO_CHAR(Purchasedate, 'MON')
ORDER BY NUMBER_OF_PURCHASES DESC;
```

Exercise_GroupBy_TotalQuantity

Summary Questions < >

Tryout 1: Exercise_GroupBy_TotalQuantity

Problem Statement

Display product id and total quantity purchased for products that have been purchased more than once. Consider only those purchase instances when the quantity purchased is more than 1.

```
HSQl
1 SELECT productid, SUM(quantitypurchased) AS "TOTAL_QUANTITY"
2 FROM Invoice
3 WHERE quantitypurchased >1
4 GROUP BY productid
5 HAVING COUNT(*) >1;
```

```
SELECT productid, SUM(quantitypurchased) AS "TOTAL_QUANTITY"
FROM Invoice
WHERE quantitypurchased >1
```

```
GROUP BY productid  
HAVING COUNT(*) >1;
```

Assignment_GroupBy_CountProduct

Tryout 1: Assignment_GroupBy_CountProduct

Problem Statement
Display the total number of products in each review category.

For the given sample data, the following is the expected output:

```
SELECT review, COUNT(*) AS "NUMBER_OF_PRODUCTS"  
FROM Product P  
GROUP BY review;
```

Assignment_GroupBy_MonthWiseCount

Tryout 1: Assignment_GroupBy_MonthWiseCount

Problem Statement
Find the number of subscribers that have subscribed in each month. Display the month as "MONTH" (column alias) and number of subscribers as "SUBSCRIBERCOUNT" (column alias).

```
SELECT TO_CHAR(startdate, 'MONTH') AS "MONTH", COUNT(*) AS "SUBSCRIBERCOUNT"  
FROM SUBSCRIPTION  
GROUP BY TO_CHAR(startdate, 'MONTH');
```

Assignment_GroupBy_TwoColumns

Tryout 1: Assignment_GroupBy_TwoColumns

Problem Statement
Retrieve the data of those subscription(s) with duration more than 3 months and the customer has got same discount percent again on another subscription. Display the Customerid, Discountpercent and average DurationinMonths of those subscriptions.

```
SELECT Customerid,discountpercent, AVG(durationinmonths) AS "AVG_DURATION"  
FROM Subscription  
WHERE durationinmonths >3  
GROUP BY Customerid, discountpercent  
HAVING COUNT(*) > 1;
```

Assignment_GroupBy_Having

Summary

Questions

Tryout > 1: Assignment_GroupBy_Having

Problem Statement

Display the name of product category(ies) for which more than one product exists.

HSQL ▾

```
1 SELECT pcategory
2 FROM Product
3 GROUP BY pcategory
4 HAVING COUNT(*)>1;
```

```
SELECT pcategory
FROM Product
GROUP BY pcategory
HAVING COUNT(*)>1;
```

Assignment_GroupBy_GenreWiseAvgMonthlyCharges

Summary

Questions

Tryout > 1: Assignment_GroupBy_GenreWiseAvgMonthlyC

A- A+

Problem Statement

For each Genre whose average monthly charge is more than 100, display the Genre and its corresponding average monthly charges as "AVG_CHARGES" (column alias).

HSQL ▾

```
1 SELECT Genre, AVG(monthlycharges) AS "AVG_CHARGES"
2 FROM Journal
3 GROUP BY Genre
4 HAVING AVG(monthlycharges)>100;
```

```
SELECT Genre, AVG(monthlycharges) AS "AVG_CHARGES"
FROM Journal
GROUP BY Genre
HAVING AVG(monthlycharges)>100;
```

Assignment_GroupBy_FilterMonthWiseCount

Summary

Questions

Tryout > 1: Assignment_GroupBy_FilterMonthWiseCount

A

Problem Statement

Identify the month(s) in which more than one subscription of duration 6 or more (months) has been taken. Display abbreviated name of month as "MONTH" (column alias) and total number of such subscriptions taken in that month as "TOTAL_SUBSCRIPTIONS" (column alias)

HSQL ▾

```
1 SELECT TO_CHAR(startdate,'MON') AS "MONTH", COUNT(*) AS "TOTAL_SUBSCRIPTIONS"
2 FROM SUBSCRIPTION
3 WHERE durationinmonths >=6
4 GROUP BY TO_CHAR(startdate,'MON')
5 HAVING COUNT(*) >1;
```

Result

```
SELECT TO_CHAR(startdate, 'MON') AS "MONTH", COUNT(*) AS "TOTAL_SUBSCRIPTIONS"
FROM SUBSCRIPTION
WHERE durationinmonths >=6
GROUP BY TO_CHAR(startdate, 'MON')
HAVING COUNT(*) >1;
```

Assignment_GroupBy_FilterAvgDiscount

Summary Questions < >

To exit full-screen, press Esc

Tryout 1: Assignment_GroupBy_FilterAvgDiscount

Problem Statement

Retrieve the JournalId and average DiscountPercent of the journals purchased before '2023-11-01' as "AVG_DIS" (column alias).

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT journalid, AVG(discountpercent) AS "AVG_DIS"  
FROM Subscription  
WHERE startdate <'2023-11-01'  
GROUP BY journalid;
```

HSQl

```
1 SELECT journalid, AVG(discountpercent) AS "AVG_DIS"  
2 FROM Subscription  
3 WHERE startdate <'2023-11-01'  
4 GROUP BY journalid;  
5
```

Result

Assignment_GroupBy_AvgMonthlyCharges

Summary Questions < >

Tryout 1: Assignment_GroupBy_AvgMonthlyCharges

Problem Statement

For every Genre except 'Trade', display the Genre and average monthly charges of the journals of that Genre as "AVERAGE CHARGE" (column alias), only if, the identified average charge(s) is less than 100.

HSQl

```
1 SELECT genre, AVG(monthlycharges) AS "Average Charge"  
2 FROM Journal  
3 WHERE genre <> 'Trade'  
4 GROUP BY genre  
5 HAVING AVG(monthlycharges)<100;
```

A- A+

```
SELECT genre, AVG(monthlycharges) AS "Average Charge"  
FROM Journal  
WHERE genre <> 'Trade'  
GROUP BY genre  
HAVING AVG(monthlycharges)<100;
```

Assignment_GroupBy_JournalWiseSubscription

Summary Questions < >

Tryout 1: Assignment_GroupBy_JournalWiseSubscription

Problem Statement

For each journal, identify the total number of subscriptions for that journal. Display JournalId and total number of subscriptions as "TOTALSUBSCRIPTIONS" (column alias).

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT journalid, COUNT(*) AS "TOTALSUBSCRIPTIONS"  
FROM Subscription  
GROUP BY journalid;
```

HSQl

```
1 SELECT journalid, COUNT(*) AS "TOTALSUBSCRIPTIONS"  
2 FROM Subscription  
3 GROUP BY journalid;
```

Result

Case 1

Assignment_GroupBy_FilterAvgCharges

Summary

Questions

Tryout > 1: Assignment_GroupBy_FilterAvgCharges

Problem Statement

Identify the publisher(s) whose average monthly charges of journals having Genre as 'Trade' or 'Politics' is less than 110. Display PublisherId and average monthly charges of the publisher as "MONTHLY_CHARGES" (column alias).

HSQL ▾

```
1 SELECT publisherid, AVG(monthlycharges) AS monthly_charges
2 FROM Journal
3 WHERE genre IN ('Trade', 'Politics')
4 GROUP BY publisherid
5 HAVING AVG(monthlycharges) < 110;
```

```
SELECT publisherid, AVG(monthlycharges) AS monthly_charges
FROM Journal
WHERE genre IN ('Trade', 'Politics')
GROUP BY publisherid
HAVING AVG(monthlycharges) < 110;
```

Assignment_GroupBy_StartDate

Summary

Questions

Tryout > 1: Assignment_GroupBy_StartDate

Problem Statement

Identify the dates on which average discount percent offered on journals that have been subscribed for atleast 6 months is more than 10. Display the identified subscription StartDate(s) and the average discount percent offered on that date as "AVG_DISCOUNT" (column alias).

HSQL ▾

```
1 SELECT startdate, AVG(discountpercent) AS "AVG_DISCOUNT"
2 FROM Subscription
3 WHERE durationinmonths>=6 AND discountpercent >10
4 GROUP BY startdate;
```

```
SELECT startdate, AVG(discountpercent) AS "AVG_DISCOUNT"
FROM Subscription
WHERE durationinmonths>=6 AND discountpercent >10
GROUP BY startdate;
```

SET OPERATION

Exercise_Union_SelectiveInvoices

Summary

Questions

Tryout > 1: Exercise_Union_SelectiveInvoices

Problem Statement

Write a query to display InvoiceID, QuantityPurchased and Discount of all products whose last digit of productid is less than 5 or that have discount less than 10%. Display duplicate records if any.

HSQL ▾

```
1 SELECT Invoiceid, quantitypurchased, discount
2 FROM Invoice
3 WHERE SUBSTR(productid, LENGTH(productid)) <5
4 UNION ALL
5 SELECT Invoiceid, quantitypurchased, discount
6 FROM Invoice
7 WHERE discount < 10;
```

```
SELECT Invoiceid, quantitypurchased, discount
FROM Invoice
WHERE SUBSTR(productid, LENGTH(productid)) <5
UNION ALL
SELECT Invoiceid, quantitypurchased, discount
FROM Invoice
WHERE discount < 10;
```

Exercise_Union_SelectiveProducts

Summary

Questions

Tryout > 1: Exercise_Union_SelectiveProducts

Problem Statement

Write a query to display Pname and Brand of the products whose Color is neither 'Brown' nor 'Yellow'. Also display the details of product(s) whose review is not 'Bad'. Do not display repeated data. Hint: Use UNION

HSQL

```
1 SELECT Pname, brand
2 FROM product
3 WHERE Color NOT IN ('Brown', 'Yellow') UNION
4 SELECT Pname, brand
5 FROM product
6 WHERE Review <> 'Bad';
```

```
SELECT Pname, brand
FROM product
WHERE Color NOT IN ('Brown', 'Yellow') UNION
SELECT Pname, brand
FROM product
WHERE Review <> 'Bad';
```

Exercise_Union_FilterVendors

Summary

Questions

Tryout > 1: Exercise_Union_FilterVendors

Problem Statement

Write a query to display the details of vendor(s) whose rating is more than 3 and contact number is known. Also display the details of vendor(s) whose name contains character 'a' and contact number is known. Hint: Use UNION ALL

HSQL

```
2 FROM Vendor
3 WHERE Rating>3 AND vendorcontactno IS NOT NULL
4 UNION ALL
5 SELECT vendorid, vendorname, vendorcontactno, rating
6 FROM Vendor
7 WHERE vendorname LIKE '%a%' AND vendorcontactno IS NOT NULL
```

```
FROM Vendor
WHERE Rating>3 AND vendorcontactno IS NOT NULL
UNION ALL
SELECT vendorid, vendorname, vendorcontactno, rating
FROM Vendor
WHERE vendorname LIKE '%a%' AND vendorcontactno IS NOT NULL
```

Assignment_CombiningData_E

Summary

Questions

Tryout > 1: Assignment_CombiningData_E

Problem Statement

Identify publishers who have not yet published any journal.

For the given sample data, the following is the expected output.

Expected Output:

HSQL

```
1 SELECT publisherid
2 FROM publisher
3 EXCEPT
4 SELECT publisherid
5 FROM journal;
```

```
SELECT publisherid
FROM publisher
EXCEPT
SELECT publisherid
FROM journal;
```

JOINS

Exercise_Join_Condition

Summary Questions < >

Tryout 1: Exercise_Join_Condition

Problem Statement

For the invoices on which a discount of more than 12% was given, display customer name, email in lower case and invoice id.

HSQL ▾

```
1 SELECT custname, LOWER(email) AS email , invoiceid
2 FROM Customer C
3 INNER JOIN Invoice I ON C.custID = I.custID
4 WHERE discount > 12;
```

```
SELECT custname, LOWER(email) AS email , invoiceid
FROM Customer C
INNER JOIN Invoice I ON C.custID = I.custID
WHERE discount > 12;
```

Exercise_Join_SelectiveCustomers

Summary Questions < >

Tryout 1: Exercise_Join_SelectiveCustomers

Problem Statement

Display ProductId as "Product Id", PName as "Product Name" and PCategory as "Category" of all the products that are purchased by either C101 or C108.

HSQL ▾

```
1 SELECT productid AS "Product Id", Pname AS "Product Name", PCategory AS "Category"
2 FROM Product P
3 INNER JOIN Invoice I ON P.productid = I.productid
4 WHERE custID IN ('C101', 'C108');
```

```
SELECT productid AS "Product Id", PName AS "Product Name", PCategory AS "Category"
FROM Product P
INNER JOIN Invoice I ON P.productid = I.productid
WHERE custID IN ('C101', 'C108');
```

Exercise_InnerJoin_SelectiveAgeGroup

Summary Questions < >

Tryout 1: Exercise_InnerJoin_SelectiveAgeGroup

Problem Statement

Display CustId as "Customer ID" and name of all customers as "Customer Name" whose age is greater than 35. Along with that display the ProductId as "Product ID" of the product(s) that the customer has purchased.

HSQL ▾

```
1 SELECT custid AS "Customer ID", custname AS "Customer Name", productid AS "Product ID"
2 FROM Customer C
3 INNER JOIN Invoice I ON I.custid = C.custid
4 WHERE age > 35 ;
```

```
SELECT custid AS "Customer ID", custname AS "Customer Name", productid AS "Product ID"
FROM Customer C
INNER JOIN Invoice I ON I.custid = C.custid
WHERE age > 35 ;
```

Exercise_Join_ConditionOnMonth

Summary Questions < >

Tryout 1: Exercise_Join_ConditionOnMonth

Problem Statement

Display customer id and customer name of all the customer(s) who have purchased any product in the month of 'August'.

Hint: A customer might have purchased products more than once in the month of August. So, duplicate rows should not appear in the result.

HSQL ▾

```
1 SELECT Custid, custname
2 FROM Customer C
3 INNER JOIN Invoice I ON I.Custid = C.Custid
4 WHERE MONTH(purchasedate) = 08
5 GROUP BY Custid,custname;
```

Result

```

SELECT Custid, custname
FROM Customer C
INNER JOIN Invoice I ON I.Custid = C.Custid
WHERE MONTH(purchasedate) = 08
GROUP BY Custid,custname;

```

Exercise_Join_CategoryWiseCount

Tryout > 1: Exercise_Join_CategoryWiseCount

Problem Statement
Identify the number of products purchased under each product category. Display the product category and number of products purchased under each category as "NUMBER_OF_PRODUCTS".

HSQl

```

1 SELECT pccategory, COUNT(*) AS "NUMBER_OF_PRODUCTS"
2 FROM Product P
3 INNER JOIN Invoice I ON I.productid = P.productid
4 GROUP BY pccategory;

```

```

SELECT pccategory, COUNT(*) AS "NUMBER_OF_PRODUCTS"
FROM Product P
INNER JOIN Invoice I ON I.productid = P.productid
GROUP BY pccategory;

```

Exercise_Join_SelectiveColorProducts

Tryout > 1: Exercise_Join_SelectiveColorProducts

Problem Statement
Identify all the White or Blue color Product(s) offered by vendors. Display vendor id, product name, color and price of identified products. Round down the price values to lower integer.

HSQl

```

1 SELECT Vendorid, Pname, color, FLOOR(price) AS baseprice
2 FROM Vendoroffering V
3 INNER JOIN Product P ON P.productid = V.productid
4 WHERE color IN ('White', 'Blue');

```

```

SELECT Vendorid, Pname, color, FLOOR(price) AS baseprice
FROM Vendoroffering V
INNER JOIN Product P ON P.productid = V.productid
WHERE color IN ('White', 'Blue');

```

Exercise_Join_CategoryWiseSelectiveCount

Tryout > 1: Exercise_Join_CategoryWiseSelectiveCount

Problem Statement
For each product category, identify the number of products with Good review and QuantityPurchased is more than 8. Display product category as "Category" and number of such products as "Total Products".

HSQl

```

1 SELECT pccategory as "Category", COUNT(*) AS "Total Products"
2 FROM Product P
3 INNER JOIN Invoice I ON I.Productid = P.Productid
4 WHERE review = "Good" AND Quantitypurchased > 8
5 GROUP_BY pccategory;

```

```

SELECT pccategory as "Category", COUNT(*) AS "Total Products"
FROM Product P
INNER JOIN Invoice I ON I.Productid = P.Productid
WHERE review = 'Good' AND Quantitypurchased > 8
GROUP BY pccategory;

```

Exercise_Join_SelectiveVendors

Summary Questions < >

Tryout 1: Exercise_Join_SelectiveVendors

Problem Statement

For the products that are offered by vendor(s) whose name is more than 15 characters long and whose price is more than 1000, display vendor name, product id and price.

```
SELECT Vendornname, productid, price  
FROM Vendoroffering Vo  
INNER JOIN Vendor V ON V.vendorid = Vo.vendorid  
WHERE LENGTH(Vendornname) > 15 AND price > 1000;
```

HSQL ▾

```
1 SELECT Vendornname, productid, price  
2 FROM Vendoroffering Vo  
3 INNER JOIN Vendor V ON V.vendorid = Vo.vendorid  
4 WHERE LENGTH(Vendornname) > 15 AND price > 1000;
```

Exercise_Join_DiscountCalculation

Summary Questions < >

Tryout 1: Exercise_Join_DiscountCalculation

Problem Statement

For every invoices in which more than 2 products were purchased, display customer id, invoice id, actual price of product as "MRP" and price applicable after applying discount as "FINAL_PRICE". Calculate actual price and final price, based on quantity and discount. Round off both the prices up to 0 decimal places.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT custid, invoiceid, round(price * quantitypurchased) AS "MRP",  
       round((price*quantitypurchased *( 1 - discount/100)),0) AS "FINAL_PRICE"  
FROM Vendoroffering VO  
INNER JOIN Invoice I ON VO.vendorid = I.vendorid AND VO.productid = I.productid  
WHERE quantitypurchased >2;
```

HSQL ▾

```
1 SELECT custid, invoiceid, round(price * quantitypurchased) AS "MRP",  
2       round((price*quantitypurchased *( 1 - discount/100)),0) AS "FINAL_PRICE"  
3 FROM Vendoroffering VO  
4 INNER JOIN Invoice I ON VO.vendorid = I.vendorid AND VO.productid = I.productid  
5 WHERE quantitypurchased >2;
```

Result

Exercise_Join_GB_OB

Summary Questions < >

Tryout 1: Exercise_Join_GB_OB

Problem Statement

Find the total number of products of each brand purchased at each location. Arrange the data in ascending order of location followed by brands in ascending order.

For the given sample data, the following rows will appear as a part of output along with other records.

```
SELECT custlocation AS "LOCATION", brand, SUM(quantitypurchased) AS "PRODUCTSSOLD"  
FROM Customer C  
INNER JOIN Invoice I ON I.custId = C.custId  
INNER JOIN Product P ON I.Productid = P.Productid  
GROUP BY custlocation, brand  
ORDER BY custlocation, brand;
```

HSQL ▾

```
1 SELECT custlocation AS "LOCATION", brand, SUM(quantitypurchased) AS "PRODUCTSSOLD"  
2 FROM Customer C  
3 INNER JOIN Invoice I ON I.custId = C.custId  
4 INNER JOIN Product P ON I.Productid = P.Productid  
5 GROUP BY custlocation, brand  
6 ORDER BY custlocation, brand;
```

Exercise_Join_CoLocatedCustomers

Summary

Questions

Tryout > 1: Exercise_Join_CoLocatedCustomers

Problem Statement

Display CustId, CustName and CustLocation of all customers who are from same location.

```
SELECT DISTINCT custid, custname, custlocation  
FROM Customer C  
INNER JOIN Customer C1 ON C1.custlocation = C.custlocation  
WHERE C.custid <> c1.custid;
```

HSQL ▾

```
1 SELECT DISTINCT custid, custname, custlocation  
2 FROM Customer C  
3 INNER JOIN Customer C1 ON C1.custlocation = C.custlocation  
4 WHERE C.custid <> c1.custid;
```

Exercise_Join_FilterProducts

Summary

Questions

Tryout > 1: Exercise_Join_FilterProducts

Problem Statement

Identify the products that belong to same category and have same review. Display PName, PCategory and Review of the identified products whose review is not bad.

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

PNAME	PCATEGORY	REVIEW
IPAD	Electronics	Bad

HSQL ▾

```
1 SELECT DISTINCT Pname, pccategory, review  
2 FROM product P  
3 INNER JOIN product P1 ON P.pcategory = P1.pcategory AND P.review = P1.review  
4 WHERE P.Pname <> P1.Pname AND review <> 'Bad';  
----- OR -----  
8 SELECT Pname, pccategory, review  
9 FROM product P  
10 INNER JOIN product P1 ON P.pcategory = P1.pcategory AND P.review = P1.review  
11 WHERE review <> 'Bad'  
12 GROUP BY Pname, pccategory, review  
13 HAVING COUNT(*) >1;
```

```
SELECT DISTINCT Pname, pccategory, review  
FROM product P  
INNER JOIN product P1 ON P.pcategory = P1.pcategory AND P.review = P1.review  
WHERE P.Pname <> P1.Pname AND review <> 'Bad';
```

----- OR -----

```
SELECT Pname, pccategory, review  
FROM product P  
INNER JOIN product P1 ON P.pcategory = P1.pcategory AND P.review = P1.review  
WHERE review <> 'Bad'  
GROUP BY Pname, pccategory, review  
HAVING COUNT(*) >1;
```

Exercise_Join_LoyalCustomers

Summary

Questions

Tryout > 1: Exercise_Join_LoyalCustomers

Problem Statement

Identify the loyal customers.

Loyal customers are those who have purchased same product from same vendor more than once.

Hint: Use self join

HSQL ▾

```
1 SELECT custid, vendorid, productid  
2 FROM Invoice I  
3 INNER JOIN Invoice II ON II.vendorid = I.vendorid  
4 AND II.productid = I.productid AND II.custid = I.custid  
5 GROUP BY custid, vendorid, productid  
6 HAVING COUNT(*) >1;
```

```

SELECT custid, vendorid, productid
FROM Invoice I
INNER JOIN Invoice I1 ON I1.vendorid = I.vendorid
AND I1.productid = I.productid AND I1.custid = I.custid
GROUP BY custid, vendorid, productid
HAVING COUNT(*) >1;

```

OUTER JOINTS

Exercise_Join_UniqueCustomerProducts

Tryout 1: Exercise_Join_UniqueCustomerProducts

Problem Statement
Display the names of all customers along with the product id of the products they have purchased. Display product id as NULL, if the customer hasn't purchased anything from the store yet.

For the given requirement, display UNIQUE records wherever applicable.

```

1 SELECT CUSTNAME,PRODUCTID
2 FROM Customer c
3 LEFT JOIN
4 Invoice I
5 on c.custId = i.custId
6 GROUP BY CUSTNAME,PRODUCTID;

```

```

SELECT CUSTNAME,PRODUCTID
FROM Customer c
LEFT JOIN
Invoice I
on c.custId = i.custId
GROUP BY CUSTNAME,PRODUCTID;

```

Exercise_Join_AllVendors

Tryout 1: Exercise_Join_AllVendors

Problem Statement
Display the names of all vendors along with the name of product offered by them. If the vendor is not offering any product currently, display product name as NULL.

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

VENDORNAME	PNAME
V1	NULL
V2	P1
V2	P2
V3	P1

```

1 SELECT VENDORNAME,PNAME
2 FROM (Vendor V
3 left join
4 VendorOffering VO
5 ON V.vendorid = VO.vendorid) Va
6 left JOIN
7 Product P
8 ON Va.productid = P.productid;

```

```

SELECT VENDORNAME,PNAME
FROM (Vendor V
left join
VendorOffering VO
ON V.vendorid = VO.vendorid) Va
left JOIN
Product P
ON Va.productid = P.productid;

```

Exercise_OuterJoin_FilterMain_W

Summary Questions

Tryout 1: Exercise_OuterJoin_FilterMain_W

Problem Statement

Identify the invoice id of all Clothing type products. If any clothing type product is not yet sold, display Invoice Id as "NOT YET SOLD".

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT productId, nvl(InvoiceId, 'NOT YET SOLD') AS "INVOICEID"
FROM Product P
left join
Invoice I
ON P.productId = I.productId
WHERE P.Pcategory = 'Clothing';
```

```
SQL
1 SELECT productId, nvl(InvoiceId, 'NOT YET SOLD') AS "INVOICEID"
2 FROM Product P
3 left join
4 Invoice I
5 ON P.productId = I.productId
6 WHERE P.Pcategory = 'Clothing';
7
```

Exercise_OuterJoin_FilterMain_O

Summary Questions

Tryout 1: Exercise_OuterJoin_FilterMain_O

Problem Statement

For all the Clothing type products that are sold, display the Invoice Id. For remaining products, display InvoiceId as "NOT YET SOLD".

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT ProductId, NVL(InvoiceId, 'NOT YET SOLD') AS "INVOICEID"
FROM Product P
left join
Invoice I
ON P.ProductId = I.ProductId and p.pcategory = 'Clothing';
```

```
SQL
1 SELECT ProductId, NVL(InvoiceId, 'NOT YET SOLD') AS "INVOICEID"
2 FROM Product P
3 left join
4 Invoice I
5 ON P.ProductId = I.ProductId and p.pcategory = 'Clothing';
6
```

Exercise_OuterJoin_FilterLookup

Summary Questions

Tryout 1: Exercise_OuterJoin_FilterLookup

Problem Statement

Display the invoice id of products that were sold after August'23. For the remaining products, display 'Not sold since Sept,23' in the invoice id column.

For the given sample data, the following records will feature as part of the output along with other record(s).

```
SELECT productid, NVL(invoiceId, 'Not sold since Sept,23') AS "INVOICEID"
FROM Product P
LEFT JOIN
Invoice I
ON P.productid = I.productid and I.PURCHASEDATE > TO_DATE('31-08-2023', 'DD-MM-YYYY');
```

```
SQL
1 SELECT productid, NVL(invoiceId, 'Not sold since Sept,23') AS "INVOICEID"
2 FROM Product P
3 LEFT JOIN
4 Invoice I
5 ON P.productid = I.productid and I.PURCHASEDATE > TO_DATE('31-08-2023', 'DD-MM-YYYY');
6
```

Assignment_Join_AndOr

Summary Questions < >

Tryout 1: Assignment_Join_AndOr

Problem Statement

Display CustomerId and CustomerName of the "Regular" or "Prime" type customers who have subscribed to any journal.

The following records will appear as a part of output along with other record(s).
Sample Output:

```
SELECT customerId, CustomerName
FROM Customer C
INNER JOIN
Subscription S
ON C.customerId = S.customerId and (customertype = 'Regular' or customertype = 'Prime')
GROUP BY customerId, CustomerName;
```

Assignment_Join_Where

Summary Questions < >

Tryout 1: Assignment_Join_Where

Problem Statement

Display JournalId and PublisherName for the journals whose publisher's contact number is available.

The following records will appear as a part of output along with other record(s).
Sample Output:

```
SELECT journalid, publishername
FROM Publisher P
INNER JOIN
Journal J
ON P.publisherid = J.publisherid
WHERE contactno IS NOT NULL;
```

Assignment_Join_CustomerWiseSubscriptionCount

Summary Questions < >

Tryout 1: Assignment_Join_CustomerWiseSubscriptionCount

Problem Statement

Identify the customer id, customer name and the total number of subscriptions as "SUBSCRIPTIONCOUNT" for the customers who have subscribed to more than one journal.

The following records will appear as a part of output along with other record(s).
Sample Output:

```
select customerId, customername, count(*) AS "SUBSCRIPTIONCOUNT"
FROM Customer C
INNER JOIN
Subscription S
ON C.customerid = S.customerid
group by customerid, customername
having count(*) >1;
```

Assignment_Join_FilterStartdate

Summary Questions < >

Tryout 1: Assignment_Join_FilterStartdate

Problem Statement

Identify the customer(s) who have subscribed for any journal on or before 1st November, 2023. Display customer name, subscribed journal name, subscription start date and the duration for the identified customers.

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
SELECT Customername, journalname, startdate, Durationinmonths
FROM Subscription S
INNER JOIN
Customer C
ON S.CustomerId = C.customerId
LEFT JOIN
Journal J
ON S.JournalId = J.journalId
WHERE startdate <= TO_DATE('01-11-2023', 'DD-MM-YYYY');
```

SQL A- A+ C G

```
1 :SELECT Customername, journalname, startdate, Durationinmonths
2 :FROM Subscription S
3 :INNER JOIN
4 :Customer C
5 :ON S.CustomerId = C.customerId
6 :LEFT JOIN
7 :Journal J
8 :ON S.JournalId = J.journalId
9 :WHERE startdate <= TO_DATE('01-11-2023', 'DD-MM-YYYY');
```

Assignment_Join_SelectiveJournals

Summary Questions < >

Tryout 1: Assignment_Join_SelectiveJournals

Problem Statement

Display journal name and genre of the Journal(s) for which more than 10% discount was given at the time of subscription.

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT JOURNALNAME, GENRE
FROM Journal J
INNER JOIN
Subscription S
ON J.JournalId = S.JournalId
WHERE S.Discountpercent >10;
```

SQL A- A+ C G

```
1 :SELECT JOURNALNAME, GENRE
2 :FROM Journal J
3 :INNER JOIN
4 :Subscription S
5 :ON J.JournalId = S.JournalId
6 :WHERE S.Discountpercent >10;
```

Assignment_Join_SameGenre_DifferentPublisher

Summary Questions < >

Tryout 1: Assignment_Join_SameGenre_DifferentPublisher

Problem Statement

Identify the publisher(s) who have published journals in same genre. Display the publishername, genre and rating for the identified publisher(s).

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
PUBLISHERNAME GENRE RATING
SELECT publishername, Genre, Rating
FROM Publisher P
INNER JOIN
Journal J
ON P.publisherId = J.publisherId
Inner join
(select genre from Journal group by genre having count(*)>1) A
On J.Genre = A.Genre
group by publishername, Genre, Rating
order by publishername,genre
```

SQL A- A+ C G

```
1 :SELECT publishername, Genre, Rating
2 :FROM Publisher P
3 :INNER JOIN
4 :Journal J
5 :ON P.publisherId = J.publisherId
6 :Inner join
7 :(select genre from Journal group by genre having count(*)>1) A
8 :On J.Genre = A.Genre
9 :group by publishername, Genre, Rating
10 :order by publishername,genre;
```

Order by publishername,Genre;

Assignment_Join_CountPublisherWiseUniqueSubscriptions

Tryout 1: Assignment_Join_CountPublisherWiseUniqueSubscriptions

Problem Statement
Display the names of publishers as "PublisherName" whose journal is subscribed by at least 3 different customers, along with numbers of unique subscribers as "NoOfUniqueSubscribers".

For the given requirements, display UNIQUE records wherever applicable.

For the given sample data, the following is the expected output.

Sample Output:

```
1 select p.publishername as "PublisherName", count(DISTINCT s.customerid) AS "NoOfUniqueSubscribers"
2 from publisher p
3 join journal j ON p.publisherid = j.publisherid
4 join subscription s ON s.journalid = j.journalid
5 group by p.publishername
6 having count(DISTINCT s.customerid) >=3;
```

Result

```
select p.publishername as "PublisherName", count(DISTINCT s.customerid) AS "NoOfUniqueSubscribers"
from publisher p
join journal j ON p.publisherid = j.publisherid
join subscription s ON s.journalid = j.journalid
group by p.publishername
having count(DISTINCT s.customerid) >=3;
```

Assignment_Join_PublisherWiseMonthlyCharges

Tryout 1: Assignment_Join_PublisherWiseMonthlyCharges

Problem Statement
Calculate total revenue generated by each publisher, based on monthly charges, subscription duration and applicable discount (if any). Display PublisherName and the calculated revenue as "TOTALREVENUE" in decreasing order of revenue generated.

For the given sample data, the following records will feature as part of the output along with other record(s).

Sample Output:

```
1 SELECT publishername, sum((monthlycharges * durationinmonths) * (1-discountpercent/100)) AS "TOTALREVENUE"
2 from publisher P
3 join Journal J ON P.PublisherId = J.PublisherId
4 join subscription S ON S.journalId = J.journalId
5 group by publishername
6 ORDER BY TOTALREVENUE DESC;
```

Result

```
SELECT publishername, sum((monthlycharges * durationinmonths) * (1-discountpercent/100))
AS "TOTALREVENUE"
from publisher P
join Journal J ON P.PublisherId = J.PublisherId
join subscription S ON S.journalId = J.journalId
group by publishername
ORDER BY TOTALREVENUE DESC;
```

Assignment_Join_CountPublisherWiseSubscriptions

Tryout 1: Assignment_Join_CountPublisherWiseSubscriptions

Problem Statement
Display the names of publishers whose journal is subscribed at least 3 times, along with numbers of subscriptions as "NO_SUBSCRIPTION".

For the given sample data, the following is the expected output.

Sample Output:

```
1 select publishername, count(*) as "NO_SUBSCRIPTION"
2 from publisher P
3 JOIN Journal J ON P.publisherid = J.publisherid
4 JOIN Subscription S ON J.journalid = S.journalid
5 group by publishername
6 having count(S.journalid)>=3;
```

```
select publishername, count(*) as "NO_SUBSCRIPTION"
FROM Publisher P
JOIN Journal J ON P.publisherid = J.publisherid
JOIN Subscription S ON J.journalid = S.journalid
group by publishername
having count(S.journalid)>=3;
```

Assignment_Join_FilterCustomers

Tryout 1: Assignment_Join_FilterCustomers

Problem Statement
Identify the customers of 'Illinois' who have subscribed 'Trade' journals.
Display customer name and subscribed journal name for the identified customers.

For the given sample data, the following records will feature as part of the output along with other record(s).

```
SQL
1 SELECT customername, journalname
2 From Customer C
3 join Subscription S ON C.customerid = S.customerID
4 join Journal J ON J.journalid = S.journalid
5 where C.state = 'Illinois' and J.genre = 'Trade';
```

```
SELECT customername, journalname
From Customer C
join Subscription S ON C.customerid = S.customerID
join Journal J ON J.journalid = S.journalid
where C.state = 'Illinois' and J.genre = 'Trade';
```

Assignment_Join_SamePublisher_SameGenre

Tryout 1: Assignment_Join_SamePublisher_SameGenre

Problem Statement
Display journal id, journal name, genre and publisher id for the journals of same genre published by same publisher.

For the given sample data, the following is the expected output:
Sample Output:

```
SQL
1 SELECT journalid, journalname, genre, PublisherId
2 From Journal J
3 Inner join
4 (SELECT genre, PublisherId from Journal group by genre, PublisherId having count(*)>=2) A
5 ON J.Genre = A.Genre and J.PublisherId = A.PublisherId;
```

```
SELECT journalid, journalname, Genre, PublisherId
From Journal J
Inner join
(SELECT Genre, PublisherId from Journal group by Genre, PublisherId having count(*)>=2) A
ON J.Genre = A.Genre and J.PublisherId = A.PublisherId;
```

Assignment_Join_CountSubscriptions

Tryout 1: Assignment_Join_CountSubscriptions

Problem Statement
For each publisher, identify genre wise total number of subscribers.
Display publisher name, genre and number of subscribers as "SUBSCRIBERCOUNT". If the publisher has not yet published a journal, display genre as "NOT PUBLISHED".
Note: All the customers might not be the subscribers.

For the given sample data, the following records will feature as part of the output along with other records(s).

```
SQL
1 SELECT publishername, NVL(genre, 'NOT PUBLISHED') AS "GENRE", count(S.Journalid) AS "SUBSCRIBERCOUNT"
2 FROM Publisher P
3 LEFT JOIN Journal J ON P.publisherId = J.publisherId
4 LEFT JOIN Subscription S ON S.Journalid = J.Journalid
5 GROUP BY publishername, GENRE;
```

```
SELECT publishername, NVL(genre, 'NOT PUBLISHED') AS "GENRE", count(S.Journalid) as "SUBSCRIBERCOUNT"
FROM Publisher P
LEFT JOIN Journal J ON P.publisherId = J.publisherId
LEFT JOIN Subscription S ON S.Journalid = J.Journalid
GROUP BY publishername, GENRE;
```

Assignment_Join_AllPublishers

Tryout 1: Assignment_Join_AllPublishers

Problem Statement
Display PublisherName, ContactNo and JournalName for all publishers. Consider JournalName as "NO JOURNAL" if the publisher has not yet published any journal.

For the given sample data, the following records will feature as part of the output along with other records).

Sample Output:

```
SELECT publishername, contactno, NVL(journalname, 'NO JOURNAL') AS "JOURNALNAME"
FROM Publisher P
LEFT JOIN Journal J ON J.publisherid = P.publisherid
```

Result

Assignment_Join_Publisher'sJournals

Tryout 1: Assignment_Join_Publisher'sJournals

Problem Statement
Display PublisherId and the Journal name as "JOURNALPUBLISHED" for the journals published by the publishers. Display "Yet To Publish" for publishers who have not published any journal.

For the given sample data, the following records will feature as part of the output along with other records).

Sample Output:

```
SELECT Publisherid, Nvl(journalName, 'Yet To Publish') AS "JOURNALPUBLISHED"
FROM Publisher P
LEFT JOIN Journal J ON J.publisherId = P.publisherId;
```

Result

Assignment_Join_FilterAcademicJournals

Tryout 1: Assignment_Join_FilterAcademicJournals

Problem Statement
For all the publishers with rating more than or same as 4, check if they have published any academic journal. If so, display publisher name along with journal name. Otherwise, display journal name as NULL.

For the given sample data, the following records will feature as part of the output along with other records).

Sample Output:

```
SELECT Publishername, journalName
FROM Publisher P
LEFT JOIN Journal J ON J.publisherId = P.publisherID and J.genre = 'Academics'
where P.rating >=4;
```

Result

```
SELECT Publishername, journalName
FROM Publisher P
LEFT JOIN Journal J ON J.publisherId = P.publisherID and J.genre = 'Academics'
where P.rating >=4;
```

SUBQUERY

Exercise_Subquery_MaxAvgDiscount

Tryout 1: Exercise_Subquery_MaxAvgDiscount

Problem Statement
Display the highest average discount (percent) offered on any product as "MAX_DISCOUNT" (column alias).

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT MAX(A.AVG_DISC) AS "MAX_DISCOUNT"
FROM (SELECT productid,AVG(discount) AS "AVG_DISC" FROM Invoice GROUP BY productid) A
```

Exercise_Subquery_MaxCount

Tryout 1: Exercise_Subquery_MaxCount

Problem Statement

Display the maximum number of purchases done by a single customer as "MAX_PURCHASES" (column alias).

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT MAX(A.noOfpurchases) AS "MAX_PURCHASES"  
FROM (SELECT custId, count(*) AS noOfpurchases FROM INVOICE GROUP BY custId) A;
```

Exercise_Subquery_MaxPurchaseDate

Tryout 1: Exercise_Subquery_MaxPurchaseDate

Problem Statement

Display the InvoiceId and PurchaseDate of most recent purchase.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT INVOICEID, PURCHASEDATE  
FROM INVOICE  
WHERE Purchasedate = (SELECT MAX(Purchasedate) FROM invoice)
```

Exercise_Subquery_AvgPrice

Tryout 1: Exercise_Subquery_AvgPrice

Problem Statement

Identify the product(s) whose average price is greater than the average price of the products offered by all the vendors. Display the ProductId and average price of those product(s) as "AVERAGEPRICE".

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT productid, AVG(price) AS "AVERAGEPRICE"  
FROM vendoroffering  
GROUP BY productid  
Having AVG(price) > (SELECT AVG(price) FROM vendoroffering);
```

Exercise_Subquery_Having

Tryout 1: Exercise_Subquery_Having

Problem Statement

List the Products whose average discount is more than average discount offered on product 'P101'.

For the given sample data, the following is the expected output.

Sample Output:

```
SELECT productId  
FROM Invoice  
Group by productid  
having AVG(discount) > (SELECT avg(discount) from invoice where productid = 'P101');
```

Exercise_Subquery_CustomerCount

Summary Questions < >

Tryout 1: Exercise_Subquery_CustomerCount

Problem Statement

Display the name of the customer(s) who have made at least two purchases.

For the given sample data, the following is the expected output.

Expected Output:

CUSTNAME

Nancy

SQL

```
-- SELECT custname
-- FROM Customer C
-- INNER JOIN Invoice I ON C.custID = I.custID
-- group by custname
-- having COUNT(*)>=2;
-- OR --
SELECT custname
FROM Customer C
WHERE C.custID IN (SELECT I.custID FROM Invoice I GROUP BY I.custID HAVING COUNT(*)>=2);
```

-- SELECT custname

-- FROM Customer C

-- INNER JOIN Invoice I ON C.custID = I.custID

-- group by custname

-- having COUNT(*)>=2;

-- OR --

SELECT custname

FROM Customer C

WHERE C.custID IN (SELECT I.custID FROM Invoice I GROUP BY I.custID HAVING COUNT(*)>=2);

Exercise_Subquery_MinTotalQty

Summary Questions < >

Tryout 1: Exercise_Subquery_MinTotalQty

Problem Statement

Display the ProductID of the product whose sale is least amongst all the products that have been purchased by any customer.

Hint: Total sale for a product would be same as total quantity purchased by any customer.

For the given sample data, the following is the expected output.

SQL

```
SELECT productID
FROM invoice
WHERE quantitypurchased = (SELECT MIN(quantitypurchased) FROM Invoice)
LIMIT 1;
```

SELECT productID

FROM invoice

WHERE quantitypurchased = (SELECT MIN(quantitypurchased) FROM Invoice)

LIMIT 1;

Exercise_GroupBy_MinTotalQuantity

Summary Questions < >

Tryout 1: Exercise_GroupBy_MinTotalQuantity

Problem Statement

Display the ProductID of the product whose sale is least amongst all the products that have been purchased by any customer.

Hint: Total sale for a product would be same as total quantity purchased by any customer.

For the given sample data, the following is the expected output.

SQL

```
SELECT productID
FROM (SELECT productID,sum(quantitypurchased) AS sale FROM Invoice GROUP BY productID ORDER BY sale)
LIMIT 1;
```

SELECT productId

FROM (SELECT productId,sum(quantitypurchased) AS sale FROM Invoice GROUP BY productId ORDER BY sale)

LIMIT 1;

Exercise_Subquery_MinTotalQuantity

Summary Questions

Tryout 1: Exercise_Subquery_MinTotalQuantity

Problem Statement

Identify the least purchased product(s). Display the product id and name of the identified product(s).

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT P.productID,P.Pname
FROM Product P
INNER JOIN
(SELECT productid,sum(quantitypurchased) AS su FROM Invoice GROUP BY productID ORDER BY su
LIMIT 1 ) A
ON P.productID = A.productID;
```

Exercise_Correlated_LeastPrice

Summary Questions

Tryout 1: Exercise_Correlated_LeastPrice

Problem Statement

For each product, identify the vendor who offers least price. Display ProductId, VendorId and Price. Arrange the records in increasing order of ProductId.

Note: Do not consider the product, if it is not yet offered by any vendor.

Hint: Different products might have same price

For the given sample data, the following is the expected output.

```
SELECT productid, vendorId, price
FROM VendorOffering
WHERE (productid,price) IN (SELECT productid, MIN(price) FROM VendorOffering GROUP BY
productid);
```

Exercise_Correlated_ThirdCostliest

Summary Questions

Tryout 1: Exercise_Correlated_ThirdCostliest

Problem Statement

Identify the third costliest product(s) offered by any vendor.

For the given sample data, the following is the expected output.

Sample Output:

VENOID	PRODUCTID	PRICE
V102	P102	1200.30

```
SELECT vendorid, productid, price
FROM VendorOffering
ORDER BY Price DESC
LIMIT 1 OFFSET 2;
```

Exercise_Correlated_Select

Tryout 1: Exercise_Correlated_Select

Problem Statement
Identify the highest discount offered to each customer of 'Yonkers'. For the given sample data, the following is the expected output.

Sample Output:

CUSTID	CUSTNAME	HIGHESTDISCOUNT
C103	Clara	15.2
C106	Peter	10.8

```
HSQl
1 SELECT custId,custName, max(discount) as "HIGHESTDISCOUNT"
2 FROM Invoice I
3 Join Customer C ON I.custId = C.custId where C.custlocation = 'Yonkers'
4 group by custId,custName;
```

```
SELECT custId,custName, max(discount) as "HIGHESTDISCOUNT"
FROM Invoice I
Join Customer C ON I.custId = C.custId where C.custlocation = 'Yonkers'
group by custId,custName;
```

SUBQUERYS

Exercise_Subquery_RatedVendor

Tryout 1: Exercise_Subquery_RatedVendor

Problem Statement
Display the names of vendors whose rating is available and have sold any product.

Note: Write the query using Correlated Subquery concept.

For the given sample data, the following is the expected output.

Expected Output:

```
HSQl
1 SELECT vendorname
2 FROM Vendor
3 WHERE vendorid IN (SELECT Vendorid FROM Invoice) AND Rating IS NOT NULL
```

```
SELECT vendorname
FROM Vendor
WHERE vendorid IN (SELECT Vendorid FROM Invoice) AND Rating IS NOT NULL
```

Exercise_Subquery_UnSoldProducts

Tryout 1: Exercise_Subquery_UnSoldProducts

Problem Statement
Display the Productid, PCategory, and brand of the product(s) that are not purchased by any customer.

Note: Write the query using Correlated Subquery concept.

For the given sample data, the following is the expected output.

Expected Output:

```
HSQl
1 SELECT productid, pCategory, brand
2 FROM Product
3 WHERE Productid NOT IN (SELECT productid FROM Invoice);
```

```
SELECT productid, pCategory, Brand
FROM Product
WHERE Productid NOT IN (SELECT productid FROM Invoice);
```

Exercise_Subquery_VendorWhoSoldProductsBefore1July

Summary Questions < >

Tryout 1: Exercise_Subquery_VendorWhoSoldProductsBefore1July

Problem Statement

Display the name of the vendor(s) who have sold at least 1 product before the month of Jul 2023.

Note: Write the query using Correlated Subquery concept.

For the given sample data, the following is the expected output:

Expected Output:

```
SELECT Vendorname  
FROM Vendor  
Where vendorid IN  
(SELECT vendorid FROM Invoice WHERE purchasedate < To_DATE('01-07-2023', 'DD-MM-YYYY'))
```

Assignment_Subquery_SubscribedJournals

Summary Questions < >

Tryout 1: Assignment_Subquery_SubscribedJournals

Problem Statement

Identify the journals which are subscribed by customers. Display JournalId, JournalName and Genre of those journals. Display unique rows.

For the given sample data, the following is the expected output:

Expected Output:

```
SELECT journalid, journalname, genre  
FROM Journal  
WHERE journalid IN (SELECT journalid FROM Subscription);
```

Assignment_Subquery_JoinstoSubquery

Summary Questions < >

Tryout 1: Assignment_Subquery_JoinstoSubquery

Problem Statement

Display JournalName and Genre of the Journals that are subscribed. For the given requirement, query using join can be written as follows:

```
SELECT DISTINCT JournalName, Genre FROM Journal J INNER JOIN  
Subscription S ON J.JournalId = S.JournalId;
```

Write a query for the same requirement using an Independent Subquery without using Joins.

```
SELECT journalName, Genre  
FROM Journal  
WHERE JournalId In (SELECT JournalId FROM Subscription);
```

Assignment_Subquery_RoundedAvgRating

Summary Questions < >

Tryout 1: Assignment_Subquery_RoundedAvgRating

Problem Statement

Display the average rating of the publishers who have published journals. The output should be rounded off to the integer value.

For the given sample data, the following is the expected output:

Expected Output:

```
SELECT ROUND(AVG(rating),0) AS "RATING"  
FROM Publisher  
WHERE Publisherid IN (SELECT Publisherid FROM Journal)
```

Assignment_Subquery_SelectiveCustomers

Summary Questions

Tryout 1: Assignment_Subquery_SelectiveCustomers

Problem Statement

Display the customer id, customer name and state of the customer(s) who reside in the same state as that of Robert. Do not display the record of Robert in the resultant set.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT Customerid, customername, C.state
FROM Customer C
WHERE C.state IN
(SELECT C1.state FROM Customer C1 WHERE customername = 'Robert')
AND customername <> 'Robert'
```

Assignment_Subquery_CustomerWith>1Subscription

Summary Questions

Tryout 1: Assignment_Subquery_CustomerWith>1Subscription

Problem Statement

Display the customer id and name of those customer(s) who have taken more than one subscriptions.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT Customerid, customername
FROM customer
WHERE Customerid IN (SELECT Customerid FROM subscription GROUP BY Customerid HAVING COUNT(*)>1);
```

Assignment_Subquery_MinMonthlyCharges

Summary Questions

Tryout 1: Assignment_Subquery_MinMonthlyCharges

Problem Statement

Identify the publisher(s) who have published a journal having minimum monthly charges across all the journals. Display the publisher id, publisher name of the identified publisher(s).

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT publisherid, publishername
FROM publisher P
INNER JOIN Journal J ON P.publisherid = J.publisherid
WHERE monthlycharges = (SELECT MIN(monthlycharges) FROM Journal);
```

Assignment_Subquery_MinMonthlyChargesOfTradeGenre

Summary Questions

Tryout 1: Assignment_Subquery_MinMonthlyChargesOfTradeGenre

Problem Statement

Identify the publisher(s) who published the journal having least Monthlycharges under 'Trade' genre. Display publisherid, publishername of those publishers.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT publisherid, publishername
FROM publisher P
```

```
INNER JOIN Journal J ON P.publisherid = J.publisherid
WHERE monthlycharges = (SELECT MIN(monthlycharges) FROM Journal WHERE genre = 'Trade');
```

Assignment_Subquery_MaxMonthlyChargesOfTradeGenre

Summary Questions

Tryout 1: Assignment_Subquery_MaxMonthlyChargesOfTradeGenre

Problem Statement

Among all the 'Trade' journals, display the JournalName and MonthlyCharges of the journal that has maximum monthly charges.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT journalname, monthlycharges
FROM Journal
WHERE monthlycharges IN
(SELECT MAX(monthlycharges) FROM Journal WHERE genre = 'Trade') AND genre = 'Trade';
```

Assignment_Subquery_CustomerFromSameStateDiffCity

Summary Questions

Tryout 1: Assignment_Subquery_CustomerFromSameStateDiffCity

Problem Statement

Identify the customers who belong to the same state but different city. Display State, City and CustomerName of the identified customers. Arrange the data in alphabetical order of state.

For the given sample data, the following is the expected output.

Expected Output:

```
SELECT C.state, city, customername
FROM Customer C
WHERE C.state IN (SELECT C1.state FROM Customer C1 GROUP BY C1.state HAVING COUNT(*)>1)
ORDER BY C.state;
```

Assignment_Subquery_SelectiveSubscriptions

Summary Questions

Tryout 1: Assignment_Subquery_SelectiveSubscriptions

Problem Statement

Identify the subscription(s) for which the discount(%) is less than or equal to the average discount(%) offered on every subscription of the same journal. Display subscription id and journal id for the identified subscription(s).

Note: Write the query using Correlated Subquery concept.

```
SELECT subscriptionid, journalid
FROM Subscription S1
WHERE discountpercent <=
(SELECT AVG(discountpercent) FROM Subscription S2 WHERE S1.journalid = S2.journalId)
```

Assignment_Subquery_CustomerSubscriptions

Tryout 1: Assignment_Subquery_CustomerSubscriptions

Problem Statement
Display customer id, customer name and subscription start date for the customer(s) who have got a discount(%) which is less than the maximum discount(%) offered on any subscription that started on the same date.

Note: Write the query using Correlated Subquery concept.

```
1: SELECT customerid, customername, startdate
2: FROM Customer C
3: INNER JOIN Subscription S ON C.customerid = S.customerid
4: WHERE Discountpercent < (SELECT MAX(Discountpercent) FROM Subscription S1 WHERE S.startdate = S1.startdate)
```

```
SELECT customerid, customername, startdate
FROM Customer C
INNER JOIN Subscription S ON C.customerid = S.customerid
WHERE Discountpercent < (SELECT MAX(Discountpercent) FROM Subscription S1 WHERE
S.startdate = S1.startdate)
```

Assignment_Subquery_Subscriptions

Tryout 1: Assignment_Subquery_Subscriptions

Problem Statement
Identify the subscription(s) on which the customers have got discount(%) more than the least discount(%) obtained by the same customer on any journal. Display CustomerName,State and Duration in months for the identified subscription(s).

For the given sample data, the following is the expected output.
Expected Output:

CUSTOMERNAME	STATE	DURATIONINMONTHS
Michael	Texas	12

```
1: SELECT customername, C.state, durationinmonths
2: FROM CUSTOMER C
3: INNER JOIN Subscription S ON C.customerid = S.customerid
4: WHERE discountpercent >
5: (SELECT MIN(discountpercent) FROM Subscription S2 WHERE C.customerid = S2.customerid)
```

```
SELECT customername, C.state, durationinmonths
FROM CUSTOMER C
INNER JOIN Subscription S ON C.customerid = S.customerid
WHERE discountpercent >
(SELECT MIN(discountpercent) FROM Subscription S2 WHERE C.customerid = S2.customerid)
```

Assignment_Subquery_CustomerWithNoSubscription

Tryout 1: Assignment_Subquery_CustomerWithNoSubscription

Problem Statement
Display the customer id and name of the customer(s) who have not subscribed for any journal. {Use Exists/Not Exists}

Note: Write the query using Correlated Subquery concept.

```
1: SELECT customerid, customername
2: FROM Customer C
3: WHERE NOT EXISTS (SELECT customerid FROM Subscription S WHERE C.customerid = S.customerid)
```

```
SELECT customerid, customername
FROM Customer C
WHERE NOT EXISTS (SELECT customerid FROM Subscription S WHERE C.customerid = S.customerid)
```

Assignment_Subquery_UnderDiscountPerCustomer

Summary

Questions

Tryout 1: Assignment_Subquery_UnderDiscountPerCustomer

Problem Statement

Identify the customer(s) who have subscribed for the journal and got the discount(%), less than the maximum discount(%) offered to all other customers on the same journal. Display CustomerName and DiscountPercent for the identified customers. Do not display duplicate records.

Note: Write the query using Correlated Subquery concept.

HSQL

```
1 SELECT DISTINCT customername, discountpercent
2 FROM Customer C
3 INNER JOIN Subscription S ON C.customerid = S.customerid
4 WHERE discountpercent <
5 (SELECT MAX(discountpercent) FROM Subscription S1
6 WHERE S.journalId = S1.journalId AND S.customerid > S1.customerid)
```

Result

Case 1

```
SELECT DISTINCT customername, discountpercent
FROM Customer C
INNER JOIN Subscription S ON C.customerid = S.customerid
WHERE discountpercent <
(SELECT MAX(discountpercent) FROM Subscription S1
WHERE S.journalId = S1.journalId AND S.customerid > S1.customerid)
```

Assignment_Subquery_MaxMonthlyChargesPerGenre

Summary

Questions

Tryout 1: Assignment_Subquery_MaxMonthlyChargesPerGenre

Problem Statement

Display the details of those journals whose monthly charge is highest in each genre. Display JournalId, JournalName, Genre and MonthlyCharges of the identified journals.

Note: Write the query using Correlated Subquery concept.

HSQL

```
1 SELECT journalid, journalname, genre, monthlycharges
2 FROM Journal
3 WHERE (genre,monthlycharges) IN
4 (SELECT genre,MAX(monthlycharges) FROM Journal GROUP BY genre);
```

```
SELECT journalid, journalname, genre, monthlycharges
FROM Journal
```

```
WHERE (genre,monthlycharges) IN
(SELECT genre,MAX(monthlycharges) FROM Journal GROUP BY genre);
```


