ANSI SQL

Clauses in SQL

LEVEL – LEARNER

Icons Used

Hands-on Exercise

Reference

Questions

Points To Ponder

Coding Standards

Lend A Hand

Summary

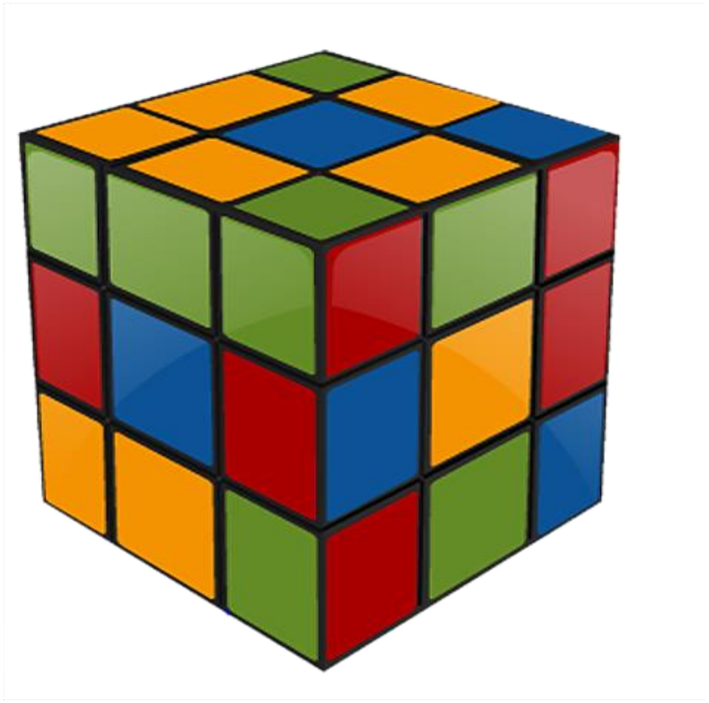
Test Your Understanding

Overview

The session on Clauses in SQL provides knowledge and understanding of the use of clauses available in ANSI. The syntax learned can be applied as part of this session in a case study provided.

Objectives

- At the end of this session, you will be able to:
  - — Identify the clauses in ANSI SQL concepts
  - — Define Group By Clause
  - — Define Having Clause
  - — Define Order By Clause



Scenario

- For the complete understanding of ANSI SQL, we are going to make use of **Product Management System** (**PMS**) for ABC Traders.
- ABC Traders is a company which buys collectable model cars, trains, trucks, buses, and ships directly from manufacturers and sells them to distributors across the globe. The above software is developed to manage the stocking, supply, and payment transactions of the firm.
- As per the requirement of the trading company, an inventory system is developed to collect the information of products and customers and their payment processing.

Database Tables

- There are many entities involved in **PMS.**
- We will be dealing with the PMS as given below throughout this course.

Offices
To maintain information of offices.
For example: Office Code, Address, City, and so on.

Customer
To maintain customer details.
For example: Customer Name and address.

Employees
To maintain employee details.
For example: ID, Name, and so on.

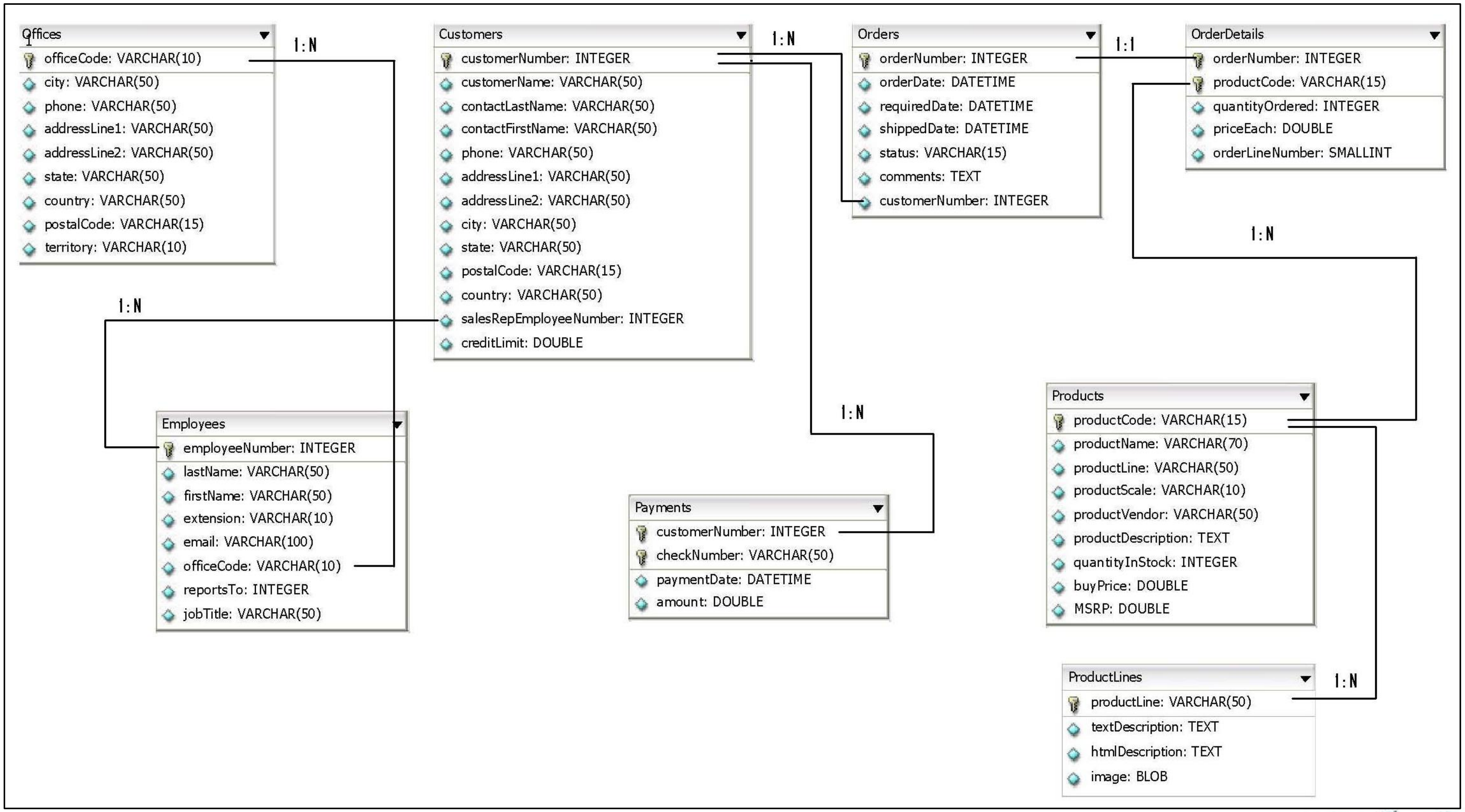
Products
To maintain information of products.
For example: Product ID, Name, and s on.

Payments
To maintain information of payments done.
For example: Payment Date, Amount, and so on.

Orders
To maintain Orders by customers.
For example: Order No., Date, and so on.

Order Details
To maintain Orders by customers.
For example: Order No., Date, and so on.

Schema Diagram



Offices
- officeCode: VARCHAR(10)
- city: VARCHAR(50)
- phone: VARCHAR(50)
- addressLine1: VARCHAR(50)
- addressLine2: VARCHAR(50)
- state: VARCHAR(50)
- country: VARCHAR(50)
- postalCode: VARCHAR(15)
- territory: VARCHAR(10)

Customers
- customerNumber: INTEGER
- customerName: VARCHAR(50)
- contactLastName: VARCHAR(50)
- contactFirstName: VARCHAR(50)
- phone: VARCHAR(50)
- addressLine1: VARCHAR(50)
- addressLine2: VARCHAR(50)
- city: VARCHAR(50)
- state: VARCHAR(50)
- postalCode: VARCHAR(15)
- country: VARCHAR(50)
- salesRepEmployeeNumber: INTEGER
- creditLimit: DOUBLE

Orders
- orderNumber: INTEGER
- orderDate: DATETIME
- requiredDate: DATETIME
- shippedDate: DATETIME
- status: VARCHAR(15)
- comments: TEXT
- customerNumber: INTEGER

OrderDetails
- orderNumber: INTEGER
- productCode: VARCHAR(15)
- quantityOrdered: INTEGER
- priceEach: DOUBLE
- orderLineNumber: SMALLINT

Employees
- employeeNumber: INTEGER
- lastName: VARCHAR(50)
- firstName: VARCHAR(50)
- extension: VARCHAR(10)
- email: VARCHAR(100)
- officeCode: VARCHAR(10)
- reportsTo: INTEGER
- jobTitle: VARCHAR(50)

Payments
- customerNumber: INTEGER
- checkNumber: VARCHAR(50)
- paymentDate: DATETIME
- amount: DOUBLE

Products
- productCode: VARCHAR(15)
- productName: VARCHAR(70)
- productLine: VARCHAR(50)
- productScale: VARCHAR(10)
- productVendor: VARCHAR(50)
- productDescription: TEXT
- quantityInStock: INTEGER
- buyPrice: DOUBLE
- MSRP: DOUBLE

ProductLines
- productLine: VARCHAR(50)
- textDescription: TEXT
- htmlDescription: TEXT
- image: BLOB

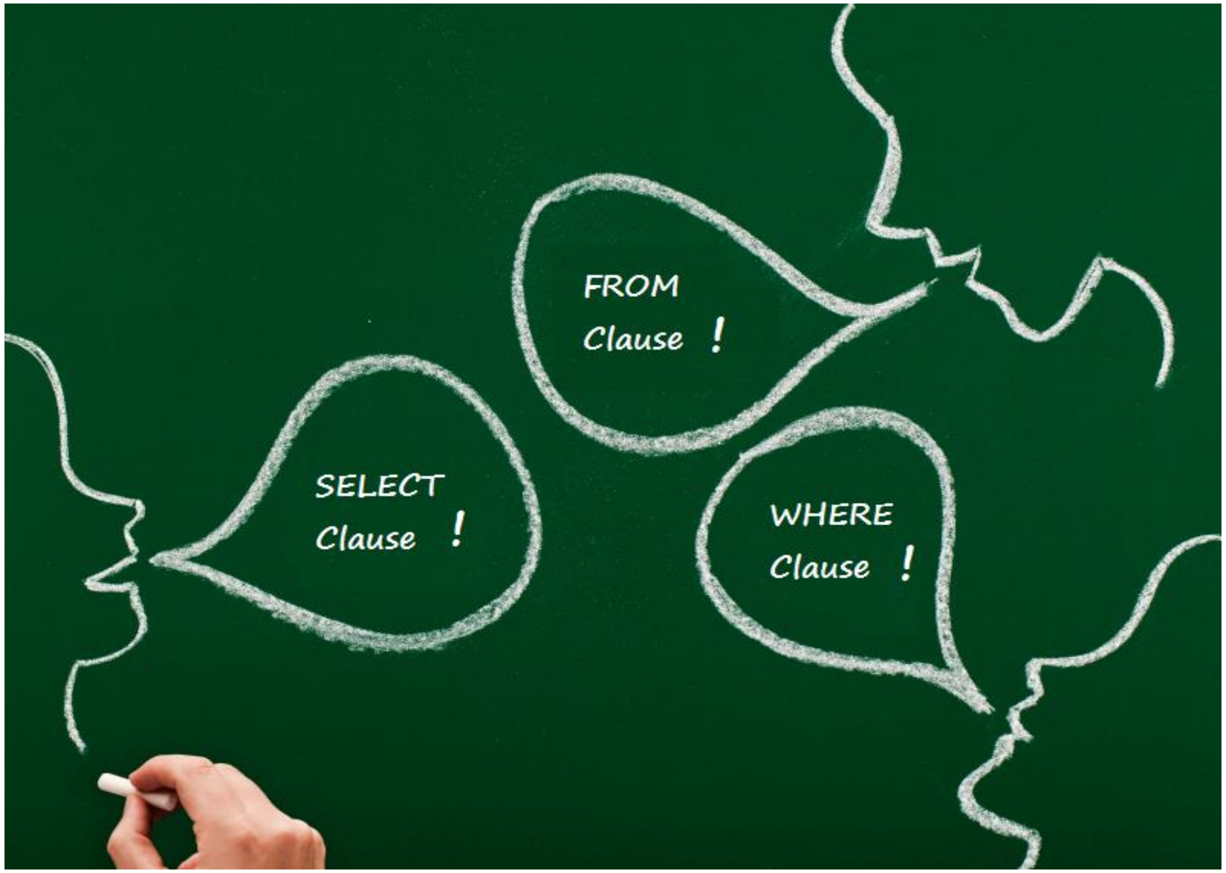
Scenario

Hi!
Now, the few requirements were implemented using operators and functions. Can you also provide the solution for finding the number of customers in each country?

Let's learn about the clauses in ANSI SQL which will help us meet Tim's requirements.

Do You Know?

- Which are the SQL Clauses that we have heard till now?
- Answer:

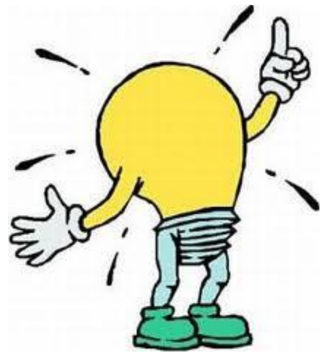


Introduction

- As we already know about SELECT, FROM and WHERE clause , let us proceed with the other three clauses namely:
  - GROUP BY
  - HAVING
  - ORDER BY

Why GROUP BY Clause?

- Why GROUP BY?
  - The GROUP BY clause is used in a SELECT statement where aggregate functions are used as one of the select fields. This is used to group the results by one or more columns specified in the select fields.
  - The SQL GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.



Rules:

- Place GROUP BY in the proper clause order
  - After the WHERE clause and before the ORDER BY clause.
- Include all non-aggregate columns in the GROUP BY clause.
- Do not use a column alias in the GROUP BY clause
  - Though table aliases are acceptable.
- NULLs are considered equivalent for grouping purposes.

What is a GROUP BY Clause?

- The GROUP BY clause is used in a SELECT statement where aggregate functions are used as one of the select fields. This is used to group the results by one or more columns specified in the select fields.

- Syntax:

```
SELECT column1, column2,  aggregate function (expression)
FROM tables
WHERE condition
GROUP BY column1, column2;
```

- The following example calculates the count of customers and displays it for each country:

```
SELECT customername, count(customername)
FROM customers
GROUP BY country;
```

Classifications of GROUP BY Clause

- Queries that return a sole value are known as *Scalar Aggregate* values.
- Scalar Aggregates do not need a GROUP BY clause.
- For example:

```
SELECT COUNT(*)
FROM CUSTOMERS;
```

- Queries that return both regular column values and aggregate functions are commonly called *Vector Aggregates*.
- Vector Aggregates use the GROUP BY clause and return one or many rows.
- For example:

```
SELECT CUSTOMERNAME, COUNTRY
FROM CUSTOMERS;
SELECT CUSTOMERNAME, COUNTRY
FROM CUSTOMERS
GROUP BY  COUNTRY;
```

Example: GROUP BY with One Select Field

CustomerNumber	Country	State
486	USA	PA
487	USA	CA
345	Japan	Tokyo
451	Japan	Tokyo
475	USA	CA
107	Japan	ANYOTHER

```
SELECT country ,
COUNT(customernumber)
FROM customers
GROUP BY country ;
```

- Output:

Country	count(customernumber)
USA	3
Japan	3

The count of customers will be calculated and displayed for each country.

Example: GROUP BY with One Column and Two Aggregate Function

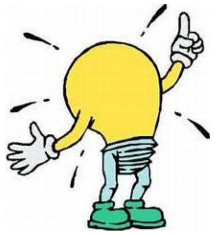
CustomerNumber	Country	State
486	USA	PA
487	USA	CA
345	Japan	Tokyo
451	Japan	Tokyo
475	USA	CA
107	Japan	ANYOTHER

```
SELECT country,
COUNT(customernumber),
COUNT(customername)
FROM customers
GROUP BY  customers;
```

- Output:

Country	Count(CustomerNumber)	Count(customername)
USA	3	3
JAPAN	3	3

The count of customer number and name will be calculated and displayed for each country.



Note: Any number of aggregate function can be used in the select field.

GROUP BY with WHERE Clause

- If WHERE clause is used along with GROUP BY clause, then WHERE clause is executed and records are selected. The group by is applied in the selected records.

- Example:

```
SELECT country, count(customernumber)
FROM customers
WHERE state != NULL
GROUP BY country;
```

Example: GROUP BY with One Select Field

- The following query displays the country and count of customers whose state is not null and is grouped by country:

CustomerNumber	Country	State
486	USA	PA
487	USA	CA
345	Japan	Tokyo
451	Japan	Tokyo
475	USA	CA
107	Japan	NULL

```
SELECT country,
COUNT(customername)
FROM customers
WHERE state!=NULL
GROUP BY country;
```

Output:

Module_id	Count(associate_id)
USA	3
JAPAN	2

The records whose state value is Not NULL will be fetched and count of customers will be calculated for each country and displayed.

Grouping One or More Columns

- How it works?
  - The records are grouped as per the first group by column. The records grouped are then further grouped by the consecutive columns. This is according to the order stated in the GROUP BY clause.

- Example:

```
SELECT Country, State, count(customernumber)
FROM customers
WHERE State != NULL
GROUP BY Country, State;
```

- The above query first groups the records by country. Consequently, the retrieved records will be grouped by state.

Example: GROUP BY with Two Columns and One Aggregate Function

CustomerNumber	Country	State
486	USA	PA
487	USA	CA
345	Japan	Tokyo
451	Japan	Tokyo
475	USA	CA
107	USA	PA

```
SELECT country, state,
COUNT(customername)
FROM customers
GROUP BY country, state;
```

- Output:

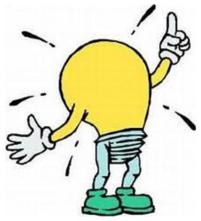
Country	State	count(customername)
USA	PA	2
USA	CA	2
JAPAN	Tokyo	2

The count of customers will be calculated for each country and state.

Why HAVING Clause?

- Why HAVING?
  - The HAVING clause adds search conditions on the result of the GROUP BY clause.
  - It does not affect the rows used to calculate the aggregates; it affects only the rows returned by the query.
  - It includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
- Syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```



Rules:
- The HAVING clause is used in combination with the GROUP BY clause.
- HAVING should not be used to eliminate rows that can be eliminated using the WHERE clause.
- HAVING conditions should always involve aggregate values.

Using Having Clause with GROUP BY

- The HAVING clause is used along with the GROUP BY clause. It is used in a SELECT statement to filter the records using the GROUP BY fields.

- **Syntax**

```
SELECT column1, column2 ,aggregate_function(column_name)
FROM table-name
WHERE condition
GROUP BY column1, column2
HAVING condition1,column2;
```

Using Having Clause with GROUP BY (Contd.)

- How it works?
  - Rows are first grouped based on GROUP BY.
  - Groups matching the HAVING clause is then displayed.

- Example: The records will be grouped based on customer name and in the group only those records which has number of customers > 2 will be displayed.

```
SELECT country, COUNT(customername)
FROM customers
WHERE state!=NULL
```

Example: HAVING Clause with GROUP BY

Without Having Clause:

```
SELECT
country,COUNT(customername)
FROM customers
WHERE state!=NULL
GROUP BY country;
```

Country	count(customername)
USA	7
UK	5
Japan	2

With Having Clause:

```
SELECT country,
COUNT(customername)
FROM customers
WHERE state!=NULL
 GROUP BY country, HAVING
count(customername)>2;
```

Country	count(customername)
USA	7
UK	5

Using Having Clause and WHERE with GROUP BY

- It is possible to use both WHERE and HAVING clauses together.
- How it works?
  - SQL first retrieves the rows based on the WHERE clause, groups the record by the group by fields and finally the grouped data are selected based on the HAVING clause.

- Example:

```
SELECT country, COUNT(customername)
FROM customers
WHERE state!=NULL
GROUP BY country, having count(customername)>2;
```

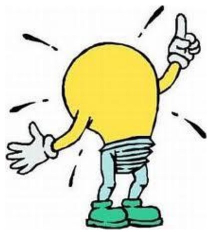
- First the records which has state not null are retrieved.
- Next, the records grouped by country are retrieved and then the grouped records are filtered for count greater than 2.

Why ORDER BY Clause?

- Why ORDER BY?
  - The ORDER BY clause allows you to sort the records in your result set.
  - It can only be used in SELECT statements.
  - A result set can be sorted through the ORDER BY clause, in accordance with the database's sort order.
  - The SQL ORDER BY clause sorts the result set based on the columns specified. If the ASC or DESC value is omitted, it is sorted by ASC.

- Syntax:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC;
```



Rules:

- The ORDER BY keyword is used to sort the result-set by **one or more** specified column(s).
- The ORDER BY keyword sorts the records in ascending order by default.
- The ordering is hierarchical, based on the order in which the items are expressed in the ORDER BY clause.
- The ordering can even be performed on columns not returned in the result set.

What is An ORDER BY Clause?

- The ORDER BY clause are used for sorting the data based on the column specified.
- Syntax:

```
SELECT columns
FROM table-name
WHERE condition
ORDER BY column  ASC/DESC;
```

- ASC indicates ascending order. (default)
- DESC indicates descending order.

- Example 1:

```
SELECT CUSTOMERNAME,COUNTRY
FROM CUSTOMERS
ORDER BY Country ASC;
```

- Return all records sorted by the country field in ascending order.

What is an ORDER BY Clause? (Contd.)

- Almost all major database vendors allow the use of ordinal positions in the ORDER BY clause.
- The order of the result set may be ordered by specifying the integer of the column_position rather than the column name or alias.
- For example :

```
SELECT au_fname, au_lname, au_id
FROM authors
ORDER BY 3, 1, 2
```

- In general, an ORDER BY clause is used to control the order of the query result set.
- If no ORDER BY clause is specified, most implementations return the data according to the physical order of data within the table or according to the order of an index utilized by the query.
- This can cause problems if the index or physical sort order of the data is changed. Instead, explicitly state the order.

Example: ORDER BY

CustomerName	Country
Atelier graphique	France
Signal Gift Stores	USA
Australian Collectors, Co.	Australia
La Rochelle Gifts	France
Baane Mini Imports	Norway
Mini Gifts Distributors Ltd	USA

Before Order By:

```
SELECT CUSTOMERNAME,COUNTRY
FROM CUSTOMERS;
```

CustomerName	Country
Australian Collectors, Co.	Australia
Atelier graphique	France
La Rochelle Gifts	France
Baane Mini Imports	Norway
Havel & Zbyszek Co	Poland
Signal Gift Stores	USA

After Order By:

```
SELECT CUSTOMERNAME,COUNTRY
FROM CUSTOMERS
ORDER BY Country ASC;
```

The rows will be sorted based on country in ascending order.

Alternate Ways of Specifying ORDER BY Clause

- Example 2:

```
SELECT CUSTOMERNAME,COUNTRY
FROM CUSTOMERS
ORDER BY 1 ASC;
```

- Where 1 is the position of the field in the select clause which is the customer name

- Example 3:

```
SELECT CUSTOMERNAME,COUNTRY
FROM CUSTOMERS
ORDER BY CustomerName DSC, Country ASC;
```

- Returns all records sorted by the supplier_city field in descending order, the sorted records are further sorted by supplier_state in ascending order.

Example: ORDER BY Two Fields

CustomerName	Country
Atelier graphique	France
Signal Gift Stores	USA
Australian Collectors, Co.	Australia
La Rochelle Gifts	France
Baane Mini Imports	Norway
Mini Gifts Distributors Ltd	USA

CustomerName	Country
Australian Collectors, Co.	Australia
Atelier graphique	France
La Rochelle Gifts	France
Baane Mini Imports	Norway
Havel & Zbyszek Co	Poland
Mini Gifts Distributors Ltd	USA

Before Order By:

```
SELECT CUSTOMERNAME, COUNTRY
FROM CUSTOMERS;
```

After Order By:

```
SELECT CUSTOMERNAME, COUNTRY
FROM CUSTOMERS
ORDER BY CustomerName DSC,
Country ASC;
```

Step 1: The records will be order based on customer name descending order
Step 2: The sorted records are sorted again on country ascending order.

Order of Execution of Clauses in SELECT Statement

- Assume that a SELECT Statement has the clauses WHERE, GROUP BY, HAVING and ORDER BY. The order of execution is as follows:
  - Selects records based on WHERE clause
  - Groups rows based on columns specified in GROUP BY clause
  - Eliminates groups based on HAVING clause condition
  - Then orders the records based on the columns specified in the order by clause

Check Your Understanding

How to order the records in ascending order?

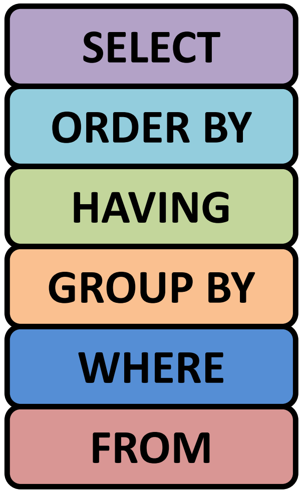
What is the difference between group and order by?

When a query has group by and order by clause what will be the order of execution precedence?

How can I filter records which are grouped?

Order of Execution

- Order of execution of clauses in SELECT Statement
  - Assume that a SELECT Statement has the FROM, WHERE, GROUP BY, HAVING and ORDER BY clause.
- The order of execution in this case will be as follows:



- Selects the Table(s) **FROM** the database.
- Selects records based on **WHERE** clause predicate.
- Groups rows based on columns specified in **GROUP BY** clause.
- Eliminates groups based on **HAVING** clause predicate.
- Sort the records based on the columns specified in the **ORDER BY** clause.
- Choose the column names specified in **SELECT** clause.

Scenario

Now we have successfully implemented few of Tim’s requirements.

Any Questions?

Lend a Hand

- Problem 1:
  - Develop a query which would retrieve the total number of students enrolled for courses on a specific date grouped by course start date and display course start date and total number of students.

- Problem 2:
  - Develop a query which would retrieve the total number of students enrolled for courses where course_type="CLR" grouped by course start date and display course start date and total number of students.

Solutions

- Solution 1:

```
SELECT SUM(NO_OF_PARTICIPANTS),COURSE_START_DATE
FROM COURSE_INFO
GROUP BY COURSE_START_DATE
```

- Solution 2:

```
SELECT SUM(NO_OF_PARTICIPANTS),COURSE_START_DATE
FROM COURSE_INFO
WHERE COURSE_TYPE='CLR'
GROUP BY COURSE_START_DATE
```

Lend a Hand

- Problem 3:
  - Develop a query which would retrieve the total number of students enrolled for courses where course_type="CLR" grouped by course start date and display course start date and total number of students where the total number of students > 10.

Solutions

- Solution 3:

```
SELECT SUM(NO_OF_PARTICIPANTS),COURSE_START_DATE
FROM COURSE_INFO
WHERE COURSE_TYPE='CLR'
GROUP BY COURSE_START_DATE
HAVING SUM(NO_OF_PARTICIPANTS)>10
```

Lend a Hand

- Problem 4
  - Develop a query which displays all the courses in increasing order of course duration.

- Problem 5
  - Develop a query which would retrieve and display the students name, their course enrolled (course name and course code), base fees. Display the records ordering the base fees in descending order.

Solutions

- Solution 4

```
SELECT COURSE_CODE, COURSE_NAME,
COURSE_START_DATE,COURSE_DURATION,
NO_OF_PARTICIPANTS,COURSE_TYPE
FROM COURSE_INFO
ORDER BY COURSE_DURATION
```

- Solution 5

```
SELECT
STUD.FIRST_NAME,COURSE.COURSE_CODE,COURSE.COURSE_NAME,F
EES.BASE_FEES FROM STUDENT_INFO STUD,COURSE_INFO
COURSE,COURSE_FEES FEES
ORDER BY FEES.BASE_FEES DESC
```

Check Your Understanding

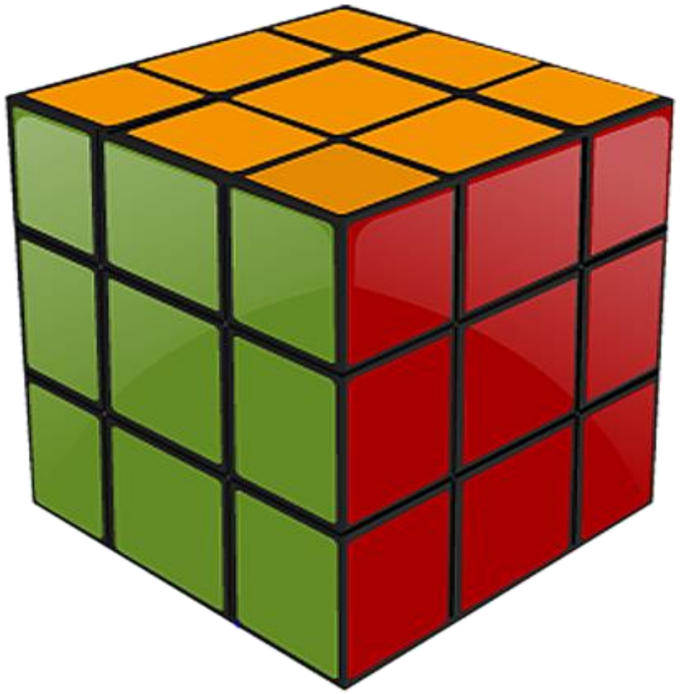
What is the use of GROUP BY clause?

What is the use of HAVING clause?

What is the use of ORDER BY clause?

Summary

- In this session, you have learned that:
  - The GROUP BY clause is used in a SELECT statement where aggregate functions are used as one of the select fields.
  - The HAVING clause is used in a SELECT statement to filter the records using the GROUP BY fields.
  - The ORDER BY clause allows to sort the records in the result set.



Source

- http://en.wikipedia.org/wiki/SQL

Change Log

Version Number	Changes made			
V1.0	Initial Version			
V1.1	Slide No.	Changed By	Effective Date	Changes Effected
	1-46	Learning Content Team CI Team CATP Technical Team	17-05-2013	Base-lining content

ANSI SQL

You have successfully completed the session on Clauses in SQL