talentech

SQL Basics

(Structured Query Language)

SQL Introduction

- The standardized query language for requesting information from a database.
- It was developed by IBM in the 1970s.
- An ANSI standard language designed for manipulation of relational databases
- A database query language widely used for accessing, querying, updating, and managing data in relational database systems

Data Definition Language

- A language used by a database management system which allows users to define the database, specifying data types, structures and constraints on the data.
- Examples are the CREATE TABLE, CREATE INDEX, ALTER, and DROP statements
- The SQL syntax used to define the way the database is physically organized.

Data Manipulation Language

- SQL commands that allow for the manipulation of data in the database, such as SELECT, INSERT, DELETE, and UPDATE
- The DML often contains features that ease report generation, including the ability to perform simple arithmetic, financial, and statistical calculations.

Tools Used

Oracle	MS SQL Server	Other
• SQL Commands • Query Builder – Oracle • SQL Plus	• Enterprise Manager • SQL Analyzer • Management Studio	• TOAD • SPUFI (DB2) • SQL Developer

6

Topic: Data Manipulation

Basic SQL

Basic SQL Statements

- **SELECT**
  - Specifies the name of the column(s) of data to retrieve.
  - All columns of a table may be retrieved using (*) a wildcard character.
- **FROM**
  - Specifies the name of the table(s) from which data is being retrieved.

Examples – SELECT & FROM

Display all employee information

```
SELECT *
FROM employees;
```

Display last name, first name, and salary

```
SELECT first_name, last_name, salary
FROM employees;
```

EMPLOYEES	
PK	EMPLOYEE_ID
	MANAGER_ID
FK1	DEPARTMENT_ID
	FIRST_NAME
	LAST_NAME
	EMAIL
	PHONE_NUMBER
	HIRE_DATE
FK2	JOB_ID
	SALARY
	COMMISION_PCT

Basic SQL Statements

- **WHERE**
  - Eliminates unwanted rows from being displayed in the result set by establishing a criteria(s).
  - More than one criteria can be specified by using conjunctive operators.
- **ORDER BY**
  - Specifies the sort order for the data retrieved.

Display last name, first name, and salary of employees make more than $7000.00

```
SELECT first_name, last_name, salary
FROM employees
WHERE Salary>7000
```

Examples – Order By

Display last name, first name, and salary of employees make more than $7000.00. Display data in ascending order by last name.

```
SELECT first_name, last_name, salary
FROM employees
WHERE Salary>7000
ORDER BY last_name
```

Display last name, first name, and salary of employees make more than $7000.00. Display the data in descending order by salary.

```
SELECT first_name, last_name, salary
FROM employees
WHERE Salary>7000
ORDER BY Salary desc
```

Order By – Multi Conditioned

- Display last name, first name, and salary of employees make more than $8000.00 and works for 100, or 110, or 130 department. Display rows by highest salary and last name.

```
SELECT first_name, last_name, salary, department_id
FROM employees
WHERE Salary > 8000 AND Department_id IN ('100', '110', '130')
ORDER BY Salary desc, last_name
```

Exercise # 1 (Basic SQL)

Code a SELECT Statement that lists all the data from Departments Table.

Code a SELECT Statement that lists Employees Last Name, First Name, Email, and Salary.

Code a SELECT Statement that lists Employees Last Name, and First Name if the Salary is more than $10,000.00

Exercise # 2 (ORDER BY)

Code a SELECT Statement that lists Employees Last Name, First Name, and Salary.

- Display the records in order by the Last name.
- Display the record by last name, and Salary.

Operators

Operator/Conditional	Description
/, –(unary), PRIOR	Positive, negative, tree traversal
*, +	Multiplication, division
/–(binary), | |	Addition, subtraction, concatenation
=, !=, >, ≳, ≲	Comparison operators
IS [NOT] NULL, LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS, IS OF	SQL-specific comparison operators
**, NOT	Exponentiation, logical negation
AND	True if both operands are true
OR	True if either operand is true
UNION, ALL, INTERSECT, MINUS	Set operators

Types of Operators

- **Comparison**
  - Establishes relationship between 2 values.
- **Logical**
  - Establishes the state of specified condition.
- **Conjunctive**
  - Allows multiple conditions to be tested.
- **Arithmetic**
  - Allows mathematical functions to be performed.

Comparison Operators

- **Equal**
  - = Exact Match
- **Not Equal**
  - <> No Match
- **Less Than**
  - < value on left is less than value on right
- **Greater Than**
  - > value on left is greater than value on right.

Example of Comparison Operators

- **Equal**
  - Where last_name = 'King'
- **Not Equal**
  - Where last_name != 'King'
- **Less Than**
  - < Where Salary < 7000.00
- **Greater Than**
  - > Where Salary >7000

Logical Operators

- **IS NULL**
  - No value exists
- **BETWEEN**
  - Retrieves values within a maximum and minimum range (includes max & min)
- **IN**
  - Retrieves values specified in a list
- **LIKE**
  - Retrieve values using similar values & wildcard characters

Example of Logical Operators

- **IS NULL**
  - Where Salary IS NULL
- **BETWEEN**
  - Where Salary Between 7000 and 8000
- **IN**
  - Where Department_Id IN (‘100’, ‘200’, ‘300’)
- **LIKE**
  - Where Last_Name LIKE ‘K%’

Conjunctive Operators

- **AND**
  - Allows multiple conditions to be tested in the WHERE clause, **all** of which must be true.
- **OR**
  - Allows multiple conditions to be tested in the WHERE clause, **any** of which may be true.

Conjunctive Operators

- **AND**
  - WHERE Department_Id = ‘100’ AND Salary> 10000
- **OR**
  - WHERE Department_Id = ‘100’ OR Salary <> 10000

Topic

Functions

Single Row Functions

- Operate on one row at a time, and return one row of output for each input row.
- Examples:
  - **CONCAT(x, y)** – Append two string and return the resulting string.
  - **LOWER() & UPPER()** – Converts letter to lower or upper character
  - **SUBSTR()** – Return substring of a string input
  - **ROUND()** – Get the result of rounding value, a whole number.
  - **DISTINCT()** – eliminate duplicate records.

Examples

- CONCAT()

```
SELECT CONCAT(first_name, last_name)
FROM Employees

Note: Use +', '+ to add space between the column values
```

- LOWER & UPPER

```
SELECT UPPER(first_name), LOWER (last_name)
FROM Employees;
```

Examples

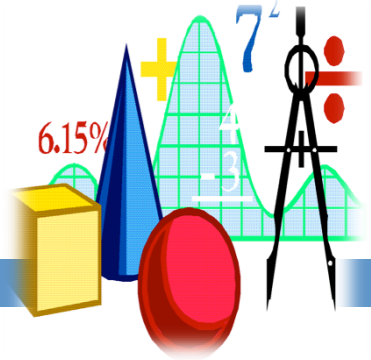
- SUBSTR()

```
SELECT SUBSTR (last_name, 0, 3)
FROM Employees
```

- DISTINCT()

```
SELECT DISTINCT(Manager_id)
FROM Employees
```

Aggregate Functions



- Count (*) or (Column Name) – Counts all the rows or only those rows with values in the specified column name.
- Max or Min (Column Name) – Returns the maximum or minimum value for the column name listed.
- SUM (Column Name) – Returns the total on the values of the specified column.
- AVG (Column Name) – Returns the Average on the values of the specified column.

Examples – Aggregate Functions

- Find the Average Salary of ALL Employees

```
SELECT AVG(Salary)
FROM Employees;
```

- Find the Total number of Employees

```
SELECT COUNT(employee_id)AS Total
FROM employees;
```

Examples – Aggregate Functions

- Find the Maximum Salary from ALL Employees.

```
SELECT MAX(Salary)
FROM employees;
```

- Find the Total Salary of ALL Employees

```
SELECT SUM(Salary)AS Total_Salary
FROM employees;
```

Exercise # 3 Functions

- Find total number of employees
- Count total number of managers
- Find the second highest salary of the company
- Display top 5 salaried employees name, phone number, and salary amount
- Display average salary of department 40 with 2 decimal point

Topic

Group By

Grouping Data

- GROUP BY –Group Identical Data; used primarily with aggregate functions.
- HAVING – Filters rows from created groups.

Examples – Group By

- Find the Average Salary of each Department.

```
SELECT Department_id, AVG(Salary)
FROM Employees
GROUP BY Department_id;
```

- Find the departments that has more than 5 employees.

```
SELECT Department_id, COUNT(employee_id)
FROM Employees
GROUP BY Department_id
Having COUNT(employee_id)>5;
```

Rules for ORDER BY and GROUP BY

- Use ORDER BY to sort; GROUP BY with aggregate functions.
- The column names(s) listed in the ORDER BY and GROUP BY clause must appear in the SELECT clause unless it is an aggregate function.
- The sort order of the ORDER BY defaults ascending, Coding DESC after the specified column names changes the sort order.

Exercise # 4 (GROUP BY)

Code a SELECT Statement that lists the departments and

- the total salary of the employees by departments.
- The total salary of the employees if more than $30,000