<u>S</u>tructured <u>Q</u>uery <u>L</u>anguage

Database & Its Manipulation

- DBMS interacts with database using a specific language
  - **SQL – Structured Query Language**
- SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems
- SQL statements are typically used to retrieve and update data in a database
- Is a **Non-Procedural Language**
- Is a **Fourth Generation Language**
  - **What to do – How to do – With What**

Structured Query Language

- Used for creating, managing, manipulating and querying the database objects such as tables, views etc.

- **Data Definition Language**
  - CREATE, ALTER, DROP, TRUNCATE
- **Data Query Language**
  - SELECT
- **Data Manipulation Language**
  - INSERT, UPDATE, DELETE
- **Data Control Language**
  - GRANT, REVOKE
- **Transaction Control Language**
  - Commit, Rollback, Savepoint

Data Definition Language (DDL)

- DDL commands are used to
  - Create database objects such as Tables, Indexes, Views, Synonyms, Sequences, Procedures, Functions, Triggers etc. in the database **CREATE**
  - Modify the structure of existing database objects **ALTER**
  - Removing the existing database objects from the database **DROP**
  - Renaming existing database objects **RENAME**
  - Adding comments to within database objects **COMMENT**
  - Describing database objects' structure. **DESCRIBE**
  - High Speed Deletion of Rows of Table **TRUNCATE**
  - Retrieving deleted table **FLASHBACK**

Creating a Table

- Factors necessary to identify for creation of table…
  - Table name
  - Column Names
  - Data Types for Columns with size
  - Constraints

Basic Data Types – Oracle

Datatype	Description	Capacity
CHAR (n)	Fixed Length Character Data of length n. Data is stored with trailing spaces	Max. of 256 Bytes
VARCHAR2 (n)	Variable Length Character Data of length n. Without any padding	Max. of 4000 Bytes
LONG	Variable Length Character Data	Max. of 2 GB
NUMBER (p,s)	Variable Length Numeric Data with precision and scale mapping.	Precision Range – 1 to 38
Date	Fixed Length Date Data Format – DD-MON-YY	Fixed 7 bytes for each row
RAW	Variable Length Binary Data	Up to 2000 bytes
%TYPE	PL / SQL Datatype	Usage Oriented
%ROWTYPE	PL / SQL Datatype	Usage Oriented

Table Creation

- **CREATE TABLE <tablename>**

```
(
      <ColumnName1  <DataType(Size)>,
      <ColumnName2  <DataType(Size)>,



     <ColumnName n  <DataType(Size)>
);
```

Table Creation

-
  ```
  CREATE TABLE  EMP
     (
          EMPNO                        INT ,
          ENAME                       VARCHAR(20),
            JOB                        VARCHAR(10),
            MGR                       INT,
            HIREDATE                   DATE,
            SAL                        DECIMAL(9,2),
          COMM                         DECIMAL(7,2),
          DEPTNO                       INT
     );
  ```

Table Creation

- **CREATE TABLE DEPT**

```
 (
     DEPTNO              INT,
     DNAME               VARCHAR(20),
        LOC              VARCHAR(10)
 );
```

Applying Column Constraints

- Constraints are part of Table Definition that are used to restrict the values entered into the columns.
- Constraints can be placed on the values while creating the table
- SQL will reject any value that violates the Constraint Criteria.
- Constraints can be added to / removed from a table after its creation and also temporarily disabled

Applying Column Constraints

NOT NULL	Prevents a column from accepting NULL Values.
UNIQUE	Ensures to avoid duplication of the values in a column.
PRIMARY KEY	Defines Key Attribute for table.
CHECK	Controls the value of a column.
DEFAULT	Assigns default value for the column when no value is specified at the time of insertion.
REFERENCES	Assigns a Foreign Key constraint to maintain Referential Integrity.

Applying Column Constraints

- CREATE TABLE EMP

```
   (
        EMPNO      INT          PRIMARY KEY,
        ENAME      VARCHAR(20)   NOT NULL,
        JOB        VARCHAR(10 ) DEFAULT 'CLERK',
        MGR        INT,
        HIREDATE   DATE,
        SAL        DECIMAL(9,2)    CHECK (SAL>0),
        COMM       DECIMAL(7,2),
        DEPTNO     INT          REFERENCES DEPT(DEPTNO)
   );
```

Applying Column Constraints

- create table dept(

```
deptno    INT,
dname     varchar(14),
loc       varchar(13),
constraint pk_dept primary key (deptno)
)
```

Inserting new Rows in a Table

- INSERT INTO Table Name

  VALUES(Value1, Value2……… Value n);

- INSERT INTO Dept

  VALUES(50, ‘HUMAN RESOURCE’, ‘PUNE’);

- INSERT INTO Dept(DeptNo, Dname)

  VALUES(50, ‘HUMAN RESOURCE’);

Exercise for Self Hunting

- Explore the DDL Statement “ALTER TABLE” with all available option and create a Text Document for it.
- Some of the suggested Operations under “ALTER TABLE”
  - Renaming a Column
  - Changing the Datatype – Size of a Column
  - Adding new Column – Removing existing Column
  - Changing – Adding – Removing the constraint of a column
  - Renaming a Table

Querying Database Tables

- SELECT Statement…
  - Instructs the database to retrieve information from table as per user's requirement

```
SELECT <column-list> FROM <table-name>
      WHERE <condition>
            GROUP BY <column-name(s)>
                  HAVING <condition>
                        ORDER BY <expression>
```

User's Requirements

Exploring SELECT

- SELECT * FROM Emp;

Displays All the columns with All the Rows

- SELECT Empno, Ename FROM Emp;
- SELECT Ename,Sal FROM Emp;
- SELECT Deptno, Empno, Mgr FROM Emp;
- SELECT Dname, loc FROM Dept;

Displays Selected Columns with All the Rows

Exploring SELECT

- Conditional Retrieval of Rows
  - WHERE Clause

No of rows to be retrieved are restricted as per condition

WHERE Clause is case sensitive.

- SELECT <column-list> FROM <table-name>

  WHERE <condition>

- SELECT Ename, Sal FROM Emp

  WHERE Sal > 1000

Exploring SELECT

List the details of the employees who have joined before end of September 81

```
SELECT  Ename, Sal, Hiredate  FROM  Emp
        WHERE hiredate <= ’30-SEP-81’ ;
```

Exploring SELECT - Special Operators

- IN :- For checking values in set.

  List the name of the employees whose employee numbers are 7369, 7521, 7839, 7788

  ```
  SELECT  Ename FROM  Emp
          WHERE empno IN (7369, 7521, 7839, 7934, 7788);
  ```

- BETWEEN :- For checking values within range.

  List the name of the employees whose salary is between 1000 and 2000

```
SELECT  Ename FROM  Emp  WHERE sal BETWEEN 1000 AND 2000 ;
```

Exploring SELECT

- Special Operators
  - LIKE :- Matching Pattern from column – Used with wild cards such as % and _

    % - Matches with zero or more occurrences of any character

    _ - Matches with one and one occurrence of any character

List Employee Names whose names start with S

```
SELECT  Ename FROM  Emp
        WHERE ename like ‘S%’ ;
```

Exploring SELECT

- List Employee Names whose names end with S

```
SELECT  Ename FROM  Emp
        WHERE ename like '%S' ;
```

- List Employee Names whose names have exactly 5 characters

```
SELECT  Ename FROM  Emp
        WHERE ename like '_____' ;
```

- List Employee Names having I as second character

```
SELECT  Ename FROM  Emp
        WHERE ename like '_I%' ;
```

Exploring SELECT

- Using Expressions with columns
  - Arithmetic Computations can be done on numeric columns
  - Alias Names can be given to Pseudo Columns

List Employee Name, Salary, Allowances (40% of Sal), P.F. (10 % of Sal) and Net Salary

```
SELECT  Ename, Sal, Sal*0.4 , Sal*0.1 , Sal + (Sal*0.4) - (Sal*0.1)
FROM  Emp ;
```

```
SELECT  Ename, Sal, Sal*0.4 “Allowance”, Sal*0.1 “PF”,
Sal + (Sal*0.4) - (Sal*0.1)  “Deductions” FROM  Emp ;
```

Exploring SELECT

- Using Expressions with columns

List Names of Employees who are more than 2 years old in organization

```
SELECT  Ename, Sal FROM  Emp
        WHERE (Year(SYSDATE()) – Year(hiredate)) > 2;
```

Exploring SELECT

- DISTINCT clause with SELECT
  - It suppresses duplicate values in column

List different jobs available in organization

```
SELECT  Job FROM  Emp ;
```

```
SELECT  Distinct Job FROM  Emp ;
```

Exploring SELECT

- Working with NULL Values
  - NULL Values are not ZEROs.
  - They are unknown or inapplicable values.
  - They can not be Used with relational / logical / arithmetic operators
  - Special operator IS is used to locate NULL values

List Employee Names who are not liable for commission.

```
SELECT  Ename FROM  Emp
        WHERE comm IS NULL;
```

List Employee Names with designations who does not report to anybody

```
SELECT  Ename, Job FROM  Emp
        WHERE mgr IS NULL;
```

Exploring SELECT

- **Ordering the Results of a Query (Data Sort)**
  - **ORDER BY Clause is used with SELECT Statement**
  - **One or more columns can specified in ORDER BY Clause.**
  - **Ordering can be done in Ascending or Descending Order. Ascending is the default order.**
  - **Order by must always be the last clause of SELECT Statement.**

List Empno, Ename and Salary in the ascending order of salary

```
SELECT  Empno, Ename, Sal FROM  Emp
          ORDER BY Sal;
```

Exploring SELECT

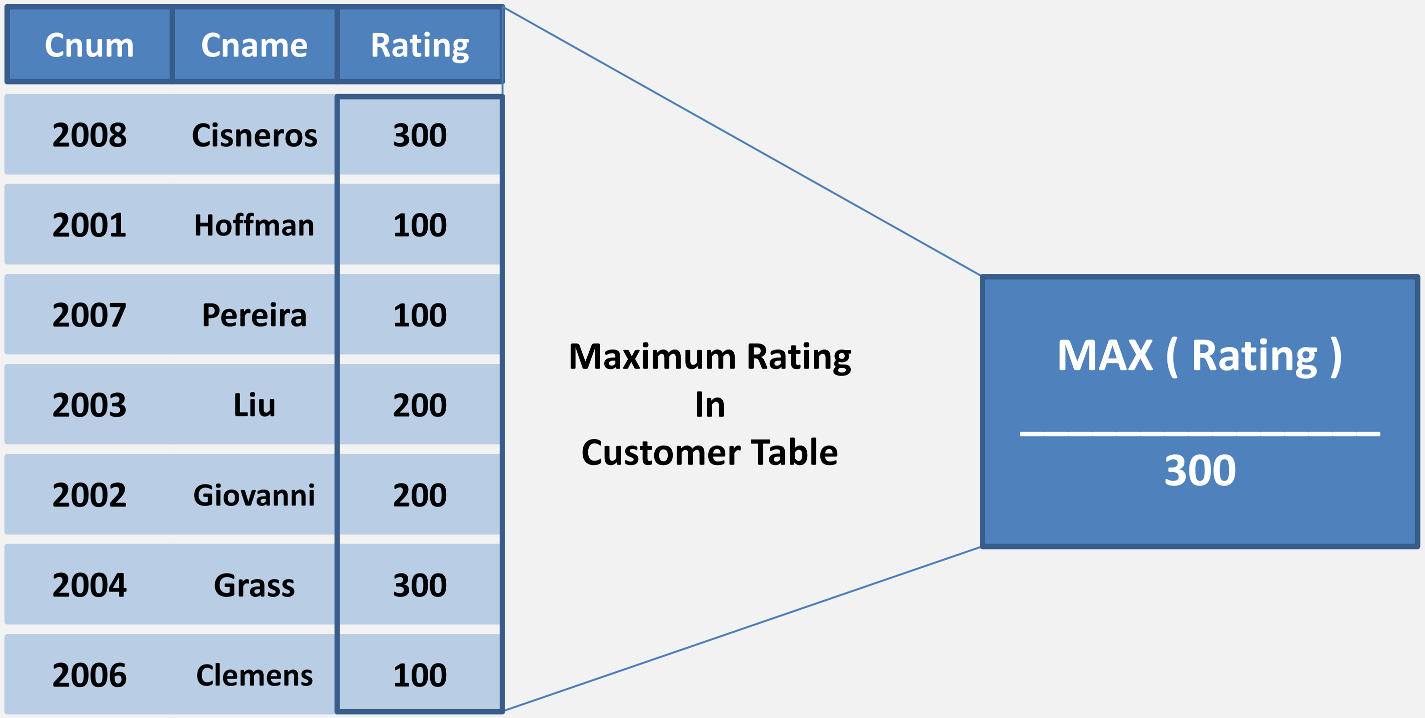
List Employee Names and Hiredate in the order who have joined latest

```
SELECT  Empno, Hiredate FROM  Emp
        ORDER BY Hiredate DESC;
```

Aggregating Data Using Group Activity

- **Group Categories**
  - A Single Group representing all the rows
    - **Default Group**
  - Multiple Groups based on specific column(s)
    - **Custom Groups**.
- **Group Activities**
  - Applying Group Functions for Aggregate Results FOR Default Grouping Strategy
  - Applying GROUP BY Clause for customized grouping
  - Applying HAVING sub - clause for conditional retrieval of groups

Find out Highest Rating in Customer Table



Cnum	Cname	Rating
2008	Cisneros	300
2001	Hoffman	100
2007	Pereira	100
2003	Liu	200
2002	Giovanni	200
2004	Grass	300
2006	Clemens	100

- **SELECT MAX (rating) FROM CUSTOMERS ;**
  - Note that Aggregate Function produces Single Row result per group.
  - And here a Default Group of all rows is considered.

Aggregate Functions

- COUNT()
  - Determines the no. of rows satisfying condition.
- SUM()
  - Determines the sum of all selected column values.
- AVG()
  - Determines the average of all selected column values.
- MAX()
  - Determines the largest of all selected column values.
- MIN()
  - Determines the smallest of all selected column values.

Exploring SELECT

Determine the total no. employees in the organization

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

How many jobs are available in the Organization ?

```
SELECT  COUNT(Distinct Job) FROM  Emp ;
```

Determine total payable salary of salesman category

```
SELECT  SUM(Sal) FROM  Emp
        WHERE Job = ‘SALESMAN’ ;
```

List the average salary and no. of employees working in department 20.

```
SELECT  AVG(Sal), COUNT(*) FROM  Emp
        WHERE Deptno = 20;
```

Customized Grouping

- Customized Group Activity can be performed by applying **GROUP BY** Clause with SELECT Statement.
  - This is achieved by segregating the rows into smaller groups.
  - Grouping can be done according to the desired column(s).
  - **HAVING** sub – clause can be used for conditional retrieval of groups.
    - What **WHERE** clause does for conditional retrieval of rows.
  - **The column name which is used for GROUPing, should appear in the column list followed by SELECT keyword.**

Find out Total Payable Salary for Each Department

- List the department numbers and total payable salary in each department.

Deptno	Sal	
10	2450	
10	5000	
10	1300	8750
20	800	
20	2975	
20	3000	
20	1100	6775
30	1600	
30	1250	2850

Total Salary of
For
Each Department

Deptno	Sum(Sal)
10	8750
20	6775
30	2850

```
SELECT    deptno,    SUM(sal)
FROM Emp      GROUP BY deptno ;
```

Customized Grouping

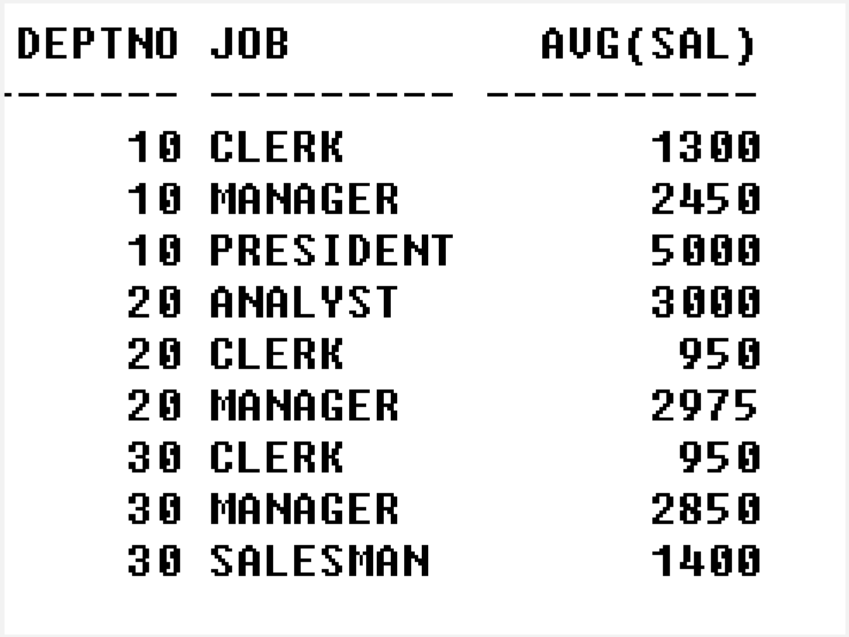
- List the department numbers and no. of employees in each department.

```
SELECT   deptno,  COUNT(*)  FROM Emp   GROUP BY deptno ;
```

Customized Grouping

- List average monthly salary for each job within each department.

```
SELECT   deptno, job, AVG(sal)   FROM Emp
                                 GROUP BY   deptno , job
                                           ORDER BY 1 ;
```



```
DEPTNO JOB          AVG(SAL)
------- --------- ----------
     10 CLERK           1300
     10 MANAGER         2450
     10 PRESIDENT       5000
     20 ANALYST         3000
     20 CLERK            950
     20 MANAGER         2975
     30 CLERK            950
     30 MANAGER         2850
     30 SALESMAN        1400
```

Conditional Retrieval of Groups

- Conditional Retrieval of Groups can be achieved by applying **HAVING Sub -** Clause followed by GROUP BY Clause.
  - The condition specified in HAVING will determine which groups to be retrieved from the total groups
- **List average salary for all departments employing more than 5 people.**

```
SELECT  deptno, AVG(Sal) FROM  Emp
           GROUP  BY Deptno
                    HAVING  COUNT(*)  > 5;
```

Exploring SELECT

- Conditional Retrieval of Rows

```
SELECT <column-list> FROM <table-name>
        WHERE <condition> ;
```

- Conditional Retrieval of Groups

```
SELECT <column-list> FROM <table-name>
        GROUP BY <column-name(s)>
                HAVING <condition> ;
```

DBMS_Assign1_PSNO1_PSNO2

1. Write a Query that that selects highest rating in each city
2. Write a Query that that selects each customer's smallest order
3. Write a Query that that counts all orders for October 10,90
4. Write a Query that that counts no. of different cities in customer
5. Write a Query that that counts no. of salespeople registering orders for each date

DBMS_Assign1_PSNO1_PSNO2

6. Write a Query that that selects largest order taken by each salesperson on each date.
7. Modify above query to select only maximum purchase amount over 3000.
8. Write a Query that that selects the first customer in alphabetical order whose name begin with G

- COMMENT ON COLUMN <*tablename*>.<*columnname*> IS 'This is a comment';

- FLASHBACK TABLE Sales1 to before DROP;

- Delete Operation
  - When you type DELETE.all the data get copied into the Rollback Tablespace first.
  - then delete operation get performed.
  - Where can be used for conditional Delete
  - Does this with DELETE overhead
  - If you envision a table, for example, as a 'flat' structure or as a series of blocks laid one after the other in a line from left to right, the high-water mark (HWM) would be the rightmost block that ever contained data

High Water Mark

- High water mark is the maximum amount of database blocks used so far by a segment. This mark cannot be reset by delete operations.
- Delete Table operation won't reset HWM.
- TRUNCATE will reset HWM.
- or example, if you delete some huge records from the database, that data will delete but the blocks are not ready to used, because that blocks are still below HWM level, so delete command never reset the HWM level,
- At the same time you truncate the data, then the data will delete and that used blocks will goto above the HWM level, now its ready to used. now they consider has free blocks.

Sequences

- **A SEQUENCE…**
  - Implicitly generates **sequential values.**
  - Can effectively be used for **auto-generation of Primary Key** Values.
  - Speeds up the efficiency of accessing sequence values when cached in memory.
  - Is a **sharable** Object.
  - Requires extremely careful handling
    - **As gaps in sequence may occur because of rollback / deletion or sharability**

Creation of Sequence

```
CREATE SEQUENCE sequence_name
                    [START WITH n]
                    [INCREMENT BY n]
                    [{MAXVALUE n | NOMAXVALUE}]
                    [{MINVALUE n | NOMINVALUE}]
                    [{CYCLE | NOCYCLE}]
                    [{CACHE n | NOCACHE}] ;
```

Use of Sequence

```
CREATE SEQUENCE Seq1
        START WITH 1
        INCREMENT BY 2
        MAXVALUE 5
        NOCACHE
        NOCYCLE;
```

```
CREATE SEQUENCE Seq2
        START WITH 1
        INCREMENT BY 2
        MAXVALUE 5
        NOCACHE
        CYCLE
```

We can check the details of sequences created

```
SELECT *  FROM user_sequences;
```

Sequence Attributes

- **CURRVAL**
  - CURRVAL obtains the current sequence value
- **NEXTVAL**
  - NEXTVAL returns the next available sequence value.
  - It returns a unique value every time it is referenced, even for different users.
- These attributes can be accessed as follows
  - sequence_name . CURRVAL or sequence_name . NEXTVAL
- **NEXTVAL must be issued for that sequence before CURRVAL contains a value.**

Using Sequence Attributes

- We can effectively use sequence attribute to generate unique values for any column of a table

```
INSERT INTO Emp1
      VALUES ( seq2 . NEXTVAL, 'SACHIN');
```

Every time we issue this statement,

A Unique value of NEXTVAL will get inserted every time.