**Rdbms (Relational –data bases management system):**

**DBMS is a software that also creation, definition and manipulation of databases. DBMS is actually a tool used to perform any kind of operation on data in databases.**

**DBMS also provided protection and security to database. IT maintains data consistency in case of multiple users.**

* **Types of DBMS:**

**DBMS is categorized into different types bases on the eay we store the data inside Database:**

1. **Flat file Data Model.**
2. **HDBMS (Hierarchical)**
3. **NBBMA (Network)**
4. **RDBMS(Relational)**

**Relational Database Concept:**

1. **‘Dr. E. F. codd’ proposed the Relational model for database system in 1970**
2. **It is basis for the Relational DBMS.**
3. **The Relational model consists of the following:**
4. **Collection of objects as Relations.**
5. **Set of operations to act on the relations.**
6. **Data integrity for accuracy & consistency. [Maintaining accuracy].**

**Constraints:**

**Constraints are the restrictions (or) conditions that are used on the columns of the table to preserve the data correctness.**

1. **Not null: Theis ensured that at least some value should be prent for an attribute. A table can gave more than one not null constraint.**

**Null: It is used to represent empty values.**

**Note: Name cannot be left null in a table.**

1. **Unique: It checks for supplicate values to unique.**

**Note: column can have multiple null values.**

**Ex. Employee ID/Mobile #Mail Id should be/could be Unique and not null.**

1. **Primary key: is used for identifying each record uniquely in a table. It is combination of NOT NULL and UNIQUE constraints. You can have only one Primary key column per table.**

**Ex.: Employee ID or Mobile # or mail-ID can be chosen as primary key.**

1. **Check: It is used for enforcing some additional condition with respect to business requirements.**

**Ex. sal > 3000, age>18.**

1. **Foreign key: It is a relational integrity constraint which creates the relationship between tae tables.**

**Foreign key can take both null and duplicate values.**

**There can be more than one foreign key for a table.**

**Ex: Dept no from DEPT table to dept no in Employee table.**

* **Avoid duplicate data.**
* **Saves space.**
* **Can have duplicate key and also null values.**

**Candidate Keys: The columns which are eligible to become primary key are knows as candidate keys.**

**Alternate Key: The columns which are eligible to become primary key but are not choosen as primary key are known as Alternate Keys.**

**PK – primary key**

**ck- candidate key**

**Sub Languages:**

|  |  |
| --- | --- |
| **Sub-Languages** | **Statements** |
| **Data Query Language (DQL)** | **SELECT** |
| **Data Definition Language** | **CREATE**  **ALTER**  **DROP**  **RENAME**  **TRUNCATE** |
| **Data Manipulation Language(DML)** | **INSERT**  **UPDATE**  **DELETE** |
| **Data Transaction Language (DTL) or Transaction Control Language. (TCL)** | **COMMETNT**  **SAVEPOINT**  **ROLLBACK** |
| **Data Control Language (DCL)** | **GRANT**  **REVOKE** |

**Sql Plus:**

**Data Query Language**

**DQL: Data Query Language, it has only one statement. i.e., SELECT. Select reads the data form the Database.**

* **‘SELECT’ has 3 capabilities.**

1. **Projection: Selecting only columns.**
2. **Selection: Selection only row.**
3. **Join: Selection rows and columns from multiple tables.**
4. **Projection:**

**Syntax for SELECT statement:**

**SELECT \* / { [DISTINCT] column/ expression[alias],… }**

**FROM table;**

**>SELECT identifies what columns.**

**>FROM identifies which table.**

**Commands in Sql**

1. **To clear the screen: CLEAR SCREEEN**
2. **To see the Structure of a table:**

**[ Structure: means column name and column data types. And structure of a table can be obtained by using the following command.]**

**DESCRIBE TABLENAME**

**ex:**

1. **User who has logged in:**

**SHOW USER**

1. **Page size in SQL Plus by default is ‘14’ and Line size is ‘80’ and we can view it by the following command:**

**SHOW PAGESIZE AND SHOW LINESIZE**

**-We can change Page size and Line Size by the following commands:**

**SET PAGESIZE and SET LINESIZE**

* **SQL commands are case insensitive and so are the SQL statements.**
* **SQL commands can be abbreviated but SQL statements cannot be abbreviated.**

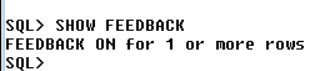
**Commands can be abbreviated but statements cannot be abbreviated:**

**SET PAGES 200 SHOW PAGES**

**SET LINES 100 SHOW LINES**

**DESC EMP**

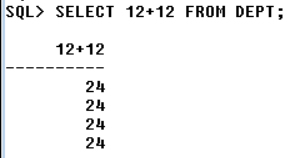
**SET FEED 1 SHOW FEED**

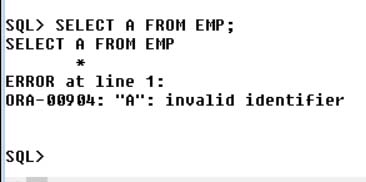
****

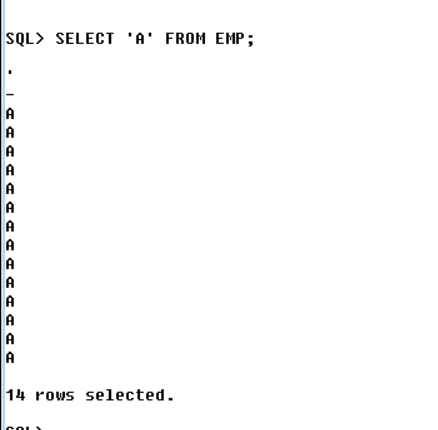
**Literal Character Strings:**

* **A literal is a character, a number, or a date included in the SELECT list.**
* **Date and Character literal values mist be enclosed within single quotation marks.**
* **Each character string is output once for each row returned.**

**e.x: SELECT 12+12 FROM DEPT;**

****

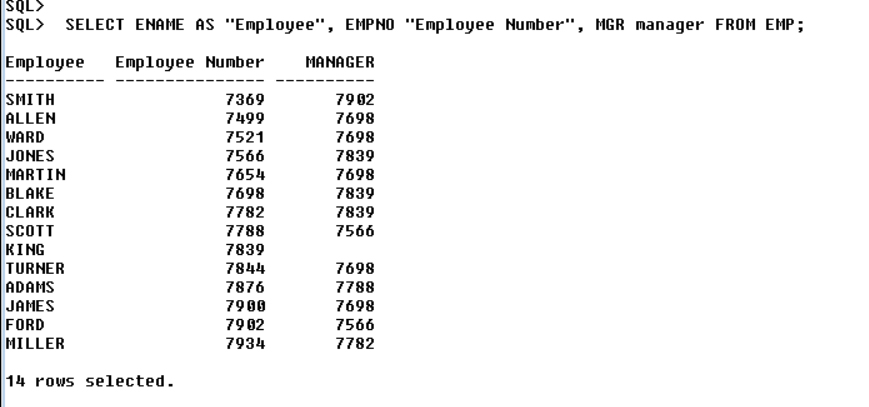
****

****

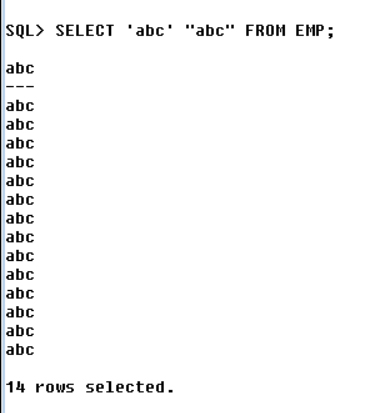
**Defining a column Alias:**

**A column alias:**

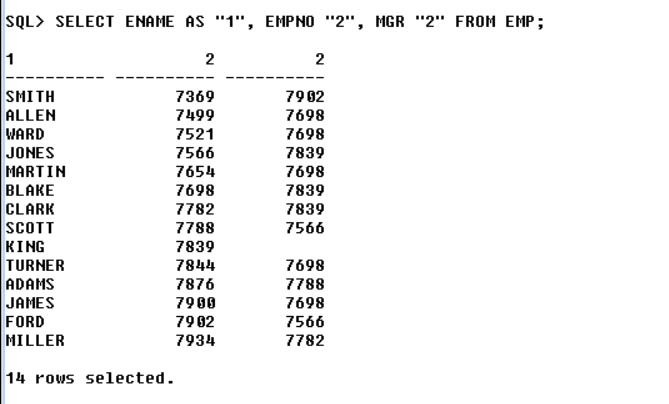
* **Renames a column heading.**
* **Is useful with calculation.**
* **Immediately follows the column name- there can also be the optional as**
* **Keyword between the column name and alias. Required double quotation marks if it contains spaces or special characters or is case sensitive.**

****

**Alias can be used with the Strings….**

****

* **We cannot have numbers as column alias, if we want to display numbers as column alias then use double quotes.**
* **We can have column alias fro literals also.**
* **We can have same column alias for more than one (or) all the columns.**

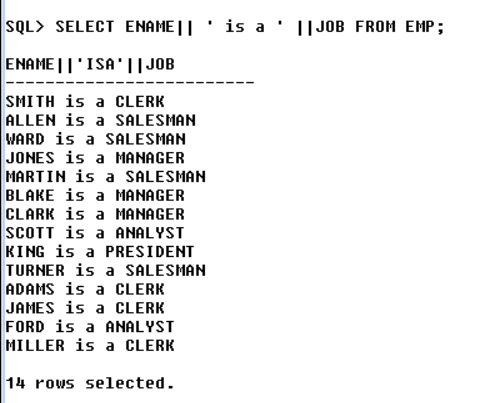
****

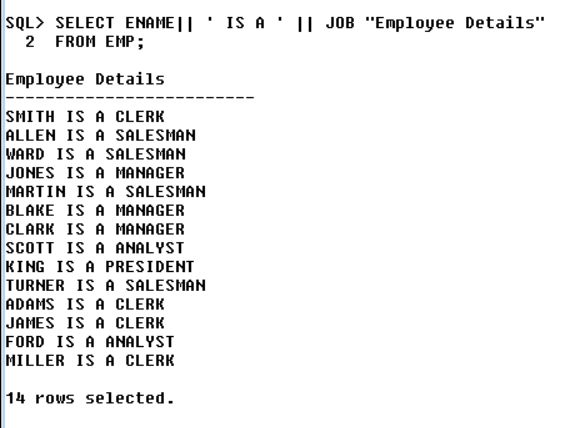
**Concatenation Operator (|| - pipe):**

**If we want to join two columns together then use concatenation operator.**

**A concatenation operator:**

* **Concatenates columns or character strings to other columns.**
* **Is represented by the vertical bars (||).**
* **Crates a resultant column that is a character expression.**

****

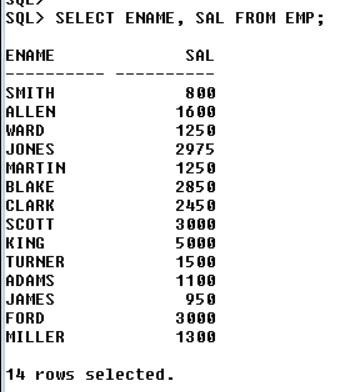
**USING ALIAS IN THE ABOVE**

* **Arithmetic Expressions**

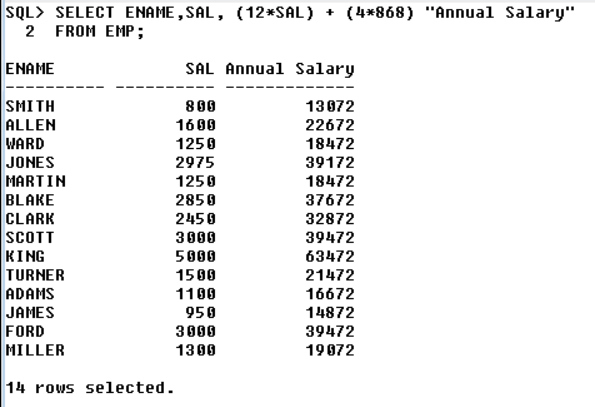
**Create expressions with number and date data by using arithmetic operators:**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **+** | **Add** |
| **-** | **Subtract** |
| **\*** | **Multiply** |
| **/** | **Divide** |

**Display Employee name with their respective salary:**

****

**Display Employee Name, Salary & Annual Salary with a quarterly commission of Rs. 868**

****

**Operator Precedence**

**\*, /, +, -**

* **Multiplication and division take priority over addition and subtraction.**
* **Operations of the same priority are evaluated from left to right.**
* **Parenthesis are used to force prioritized evaluation and to clarify statements.**

**Defining a Null value:**

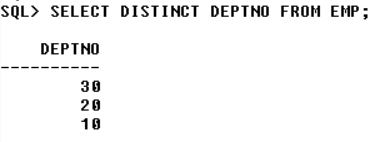
* **A null value is a value that is unavailable, unassigned, unknown or inappropriate.**
* **A null is not as same as zero or a blank space.**
* **Arithmetic Expression contains a null value evaluated to null.**

**The ‘Distinct’ keyword**

**It is used to display only the distinct (different ) record.**

**SELECT DISTINCT DEPNO**

**FROM EMP;**

****

**SQL \* Plus / SQL Plus**

**Client / Tool to access the Database.**

**Installed along with Database.**

**Has Specifies set of Commands.**

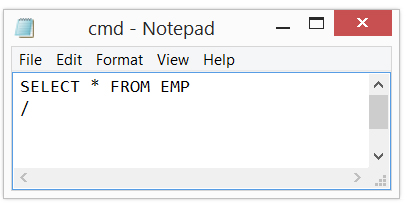
**Interface between User and Database.**

**Commands to save files:**

**Save: Saves the contents of the SQL buffer in a host operating system file.**

**e.g: Untitled-1.jpg**

**Creates a file and stores Recent Previous Command executed.**

****

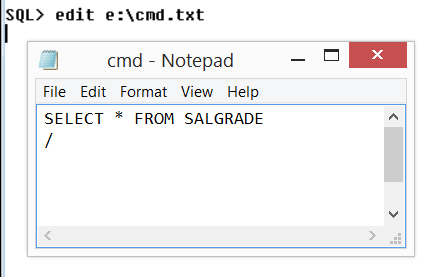
**append: If there is a file which is already created, the append keyword will add the command line in the file.**

**e.g: Untitled-1.jpg**

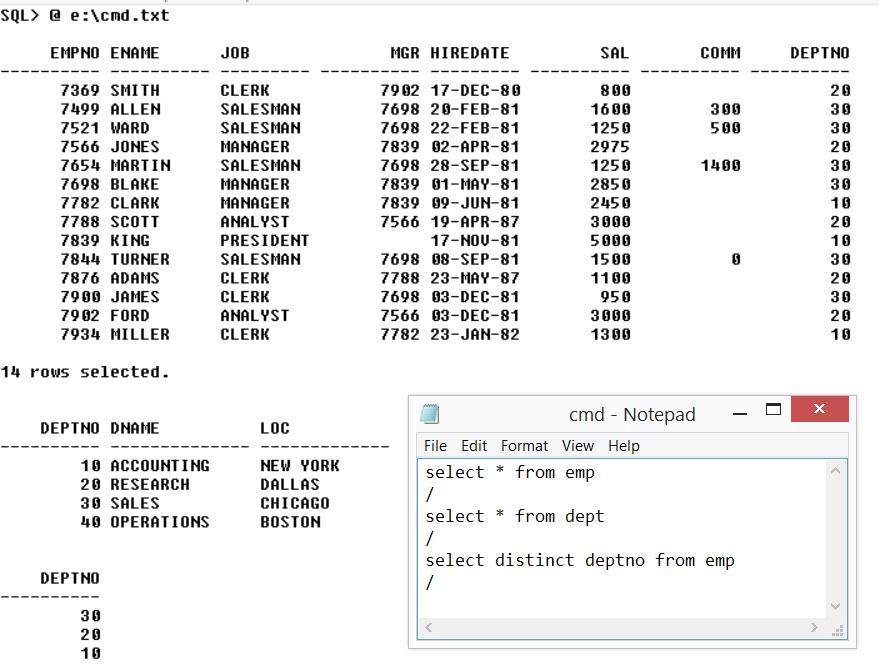
**replace: This command will clear the file contents which is already created and replace the recent command.**

**e.g: Untitled-1.jpg**

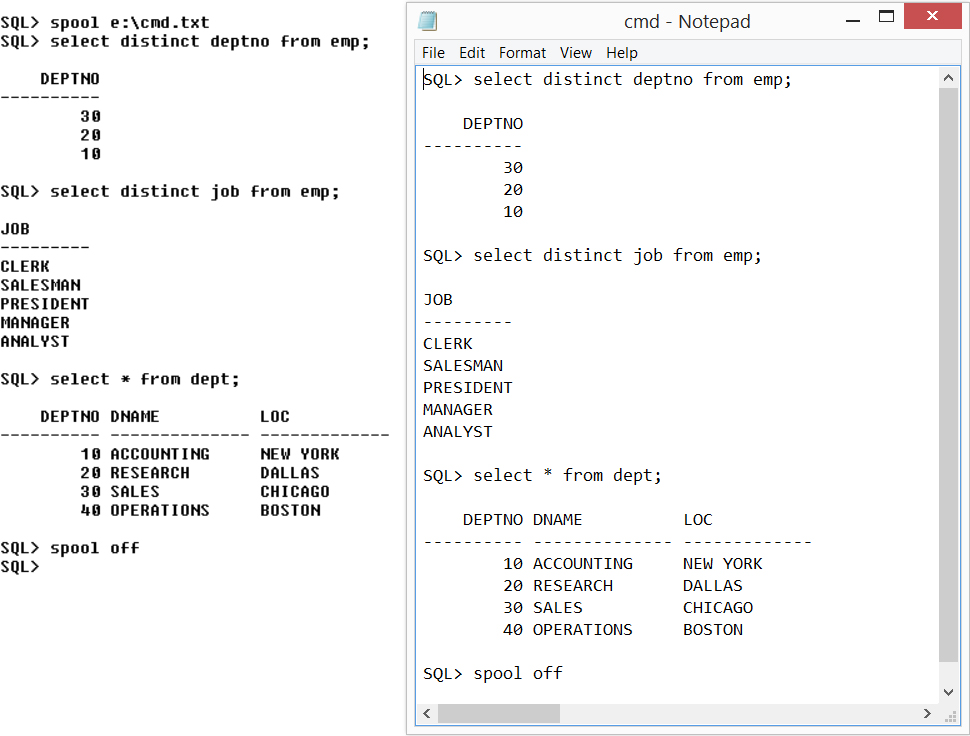
**edit: is used to edit the contents of the saved file . It will open the text editor.**

**e.g: **

**@: This will show the output for the commands which are saved in the file, it will execute the statements one after one.**

**e.g: **

**Spool: Record every result in an operating System file and optionally, sends the file to a printer. You need to mark the spool off by command: spool off**

**e.g: **

**Selection**

**Limiting the rows selected.**

**Restrict the rows returned by using the where clause.**

**Syntax: SELECT \* / { [DISTINCT] COLUMN / EXPRESSION [ ALIAS]}**

**FROM tablename [WHERE CONDITION(S)];**

**Comparison Conditions**

|  |  |
| --- | --- |
| Operator | Meaning |
| = | Equals to |
| > | Greater than |
| >= | Greater than or equals to |
| <= | Less than or equals to |
| <>, != | Not equals to |

SQL statements versus Sql \* plus commands:

|  |  |
| --- | --- |
| SQL | SQL \* Plus |
| 1. A language. | 1. An Environment. |
| 1. ANSI standard. | 1. Oracle proprietary |
| 1. Keywords cannot be abbreviated | 1. Keywords can be abbreviated |
| 1. Statements manipulate data and table definitions in the database | 1. Commands do not allow manipulation of values in the database. |
| 1. SQL Statements | 1. SQL \* plus commands. |

Q: Display the employees who belong to 30th department?

A: **select \* from emp where deptno = 30;**

Q: Display employee name, salary, job and hire date of those employees who salary is between 1260 and 3010?

A: **select \* from emp where sal>=1260 and sal<=3010;**

Or

**select \* from emp where sal BETWEEN 1260 AND 3010;**

**syntax:** WHERE columnName BETWEEN lessValue AND greaterValue;

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| BETWEEN  …AND… | Between two values  (inclusive) |
| IN(set) | Match any of a list of values |
| LIKE | Match any of a list of values |
| IS NULL | Is a null value |

Q: Display the employees whose employee number is lying between 7499 and 7876?

A: **select \* from emp where empno between 7499 and 7876;**

Q: Display all the mangers and analyst?

A: **select \* from emp where job in ('MANAGER','ANALYST');**

Q: Display the employees who belong to 10th or 30th department.

A: **SELECT \* FROM EMP WHERE DEPTNO IN (10,30);**

Q: Display the employees who are earning a salary of rupees 1250 or 5000?

A: **SELECT \* FROM EMP WHERE SAL IN (1250,5000);**

Q: Display those employees whose employee number is 7369, 7639, 7839, 7788, 7566?

A: **SELECT \* FROM EMP WHERE EMPNO IN(7369,7639, 7839, 7788, 7566);**

Q: Display the employee whose name is beginning with ‘A’?

A: **SELECT \* FROM EMP WHERE ENAME LIKE 'A%';**

LIKE operator is like a wildcard Operator.

Q: Display the employee whose name is ending with N?

A: **SELECT \* FROM EMP WHERE ENAME LIKE '%N';**

Q: Display the employee whose name is having ‘E’ as the last but one character or ‘E’ as the second last character?

A: **SELECT \* FROM EMP WHERE ENAME LIKE '%E\_';**

Q: Display the employees whose name has exactly four characters?

A: **SELECT \* FROM EMP WHERE ENAME LIKE '\_\_\_\_';**

Q: Display the employees whose name has at least two ‘LL’?

A: **SELECT \* FROM EMP WHERE ENAME LIKE '%L%L%';**

Q: Display all the ‘man’? OR Display the employees whose job has ‘Man’ as a string.

A: **SELECT \* FROM EMP WHERE JOB LIKE '%MAN%';**

Q: Display the employees who do not have a manager?

A: **SELECT \* FROM EMP WHERE MGR IS NULL;**

**This is specially used with the null values, not all the null values are equal in oracle. Remember ‘0’ (zero) is not equal to null.**

Q:Display the employees who are not earning any commission.

A: **SELECT \* FROM EMP WHERE COMM IS NULL OR COMM = 0;**

**Using the LIKE condition:**

Use the LIKE condition to perform wildcard searches of a wildcard string values.

Search conditions can contain either literal characters or numbers:

% = denotes zero or many characters.

\_ =denotes one characters.

**Logical Conditions:**

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| **AND** | Returns TRUE if both component conditions are true. |
| **OR** | Returns TRUE if either conditions is true. |
| **NOT** | Returns TRUE if the following condition is FALSE. |

Q: Display the employee who are earning more than 1500 with their designation either as manage or analyst.

A: **SELECT \* FROM EMP WHERE SAL > 1500 AND JOB IN (‘MANAGER’,’ANALYST’);**

Q: Display the employees whose salary is more than 1500 per month with their designation neither as Manager nor as Analyst.

A: **SELECT \* FROM EMP WHERE SAL > 1500 AND JOB NOT**

**IN('MANAGER','ANALYST');**

Q: Display all the salesman whose salary is between 1000 and 2500, name beginning with ‘A’;

A: **SELECT \* FROM EMP WHERE**

**JOB=’SALESMAN’**

**AND SAL BETWEEN 1000 AND 2500**

**AND ENAME LIKE 'A%';**

Q: Display the employee whose name is not beginning with ‘A’ and are earning some commission?

A: **SELECT \* FROM EMP WHERE ENAME NOT LIKE 'A%' AND COMM > 0;**

Q: Display the employees whose designation either manager or clerk or salesman, name not ending with ‘N’?

A: **SELECT \* FROM EMP WHERE JOB IN (‘MANAGER’,’CLERK’,’SALESMAN’) AND NOT ENAME LIKE ‘%N’;**

Q: Display the employees whose salary is between 1500 and 5000 , earning some commission with designation neither clerk nor analyst, joined after 1980?

A: **SELECT \* FROM EMP WHERE SAL BETWEEN 1500 AND 5000**

**AND COMM > 0**

**AND NOT JOB IN('CLERK','ANALYST')**

**AND HIREDATE > '31-DEC-80';**

**Rules of Precedence:**

|  |  |
| --- | --- |
| 1 | Arithmetic Operators |
| 2 | Concatenation Operators |
| 3 | Comparison conditions |
| 4 | IS NULL, LIKE, IN |
| 5 | BETWEEN |
| 6 | NOT legal condition |
| 7 | AND logical condition |
| 8 | OR logical condition |

We can override the rules of precedence by using parenthesis

**‘ORDER BY’ clause:**

syntax:

SELECT\*/{[DISTICNCT] column/expression[alias],…}

FROM table

[WHERE CONDITION(S)]

[ORDER BY {column, expression, alias,} [ASC?DESC]];

example:

**SELECT \* FROM EMP ORDER BY SAL;**

**SELECT \* FROM EMP ORDER BY SAL DESC;**

**SELECT ENAME, SAL FROM EMP ORDER BY SAL,ENAME; (**If there is any ambiguity in the first column, than it will go to the next Colum to arrange the ambiguous column**)**.

Q: Display company owner information from emp table?

A:

Q: List all the even salaries from the emp table?

A:

For Selection

Q: Display the employees who are getting 2500 and excess salaries in the department 20?

A:

Q: Display all the managers working in 20 and 30 dept?

A:

Q: Display all the employees whose job is manger, who don’t have a manager?

Q: Display all the employees who are getting some commission with their designation is neither MANAGER nor ANALYST?

A:

Q: Display all the employees whose naming is having letter ‘E’ as the last but one character?

A:

Q: Display all the Employees whose total salary is more than 2000?

A:

Q: Display all the employees who are getting some commission in department 20 and 30?

A:

Q: Display all the managers whose name doesn’t start with ‘A’ and ‘S’?

A:

Q: Display all the employees whose earning salary not in the range of 2500 and 5000 in dept 10 and 20.

A:

Q: Display all the employees whose name has exactly 4 characters?

A:

Q: Display all the employees whose annual salary is ending with 0?

A:

Q: List all the even salaries?

A:

Q: List all the odd Salaries?

A:

Q: Display company owner’s information?

A:

Q: Display all the employees who are SALESMAN having ‘E’ as the last but one characters in ename but with salary having exactly 4 characters?

A:

Q: Display all the employees who have joined after year 81?

A:

Q: Display all the employees who have joined in Feb?

A:

**Functions**

Functions are re-usable programs

Function may or may not take input but they always return an output.

**SQL Functions**

(Output)

**Function**

**arg 1**

**Result Value**

**arg 2**

**arg 3**

Functions are very power feature of SQL and can be used to do the following:

* Perform calculation on data.
* Modify individual data items.
* Manipulate output for groups of rows.
* Format dates and numbers for display.
* Convert column data types.
* SQL functions sometimes take arguments and always return a value. Functions are reusable programs.
* Two Types of SQL Function

**Functions**

**I/P (Greater than one row)**

**O/P**

**I/P**

**O/P**

**Multiple Row Function**

**Single – row Function**

Q: Display the employee whose name is begging with a using substring function?

A: **SELECT ENAME, SUBSTR (ENAME,1,1) FROM EMP WHERE SUBSTR(ENAME,1,1)=’A’;**

**Q:** Display the employees whose name is ending with ‘N’ ?

A: **SELECT \* FROM EMM WHERE SUBSTR(ENAME,-1,1)=’N’;**

**Q:** Display the employees whose name is having ‘E’ as the last but one character?

**A: SELECT \* FROM EMP WHERE SUBSTR(ENAME, -2, 1)=’E’;**

**Iterator**

**SELECT ENAME SUBSTR(ENAME, 3, 1), SUBSTR(ENAME,3) FROM EMP;**

**INSTR(‘JSPIDERS’, ‘S’, 1, 2)**

1st arg- Main string

2nd arg- search String

3rd- Position

4th- No. of Occurences

‘J S P I D E R S’

1 2 3 4 5 6 7 8

If we put positive integer in the argument it will iterate towards right.

‘J S P I D E R S’

-8 -7 -6 -5 -4 -3 -2 -1

If we put negative integers the iteration will start from left.

**SELECT ‘JSPIDERS’, INSERT(‘JSPIDERS’, ‘S’, 1, 2) FROM DUAL;**

**SELECT ‘JSPIDERS’, INSERT(‘JSPIDERS’, ‘S’, 1, 3) FROM DUAL;**

**O/P: 0**

Q: Display the position of the first occurrence of ‘A’ in all the ENAME?

A: **SELECT INSTR,(ENAME,’A’,1,1) FROM EMP;**

**Q:** Display those employees whose name is beginning with ‘A’ using INSTR()?

A:  **SELECT ENAME, INSTR(ENAME,’A’,1,1) FROM EMP WHERE INSTR(ENAME,’A’,1,1) = 1;**

**Q:** Employees whose name is ending with ‘N’?

A: **SELECT ENAME, INSTR(ENAME, -1, 1)= LENGTH(ENAME);**

Q: Display the employees whose name has ‘E’ as the last but 1st character?

A: **SELECT ENAME, INSTR(ENAME,’E’, -2, 1) WHERE INSTR(ENAME, ‘E’, -2, 1)=LENGTH(ENAME) - 1;**

**SELECT 'JSPIDERS' , INSTR('JSPIDERS','S',1,1),INSTR('JSPIDERS','S') FROM DUAL;** (If we don’t give the 3rd and 4th arguments in the INSTR, it will automatically take those arguments as 1)

**NESTING Functions:**

1. Single-row functions can be nested to any level.
2. Nested functions are evaluated from deepest level to deep level.

|  |
| --- |
| F3 ( F2 ( F1 ( col , arg1 ) , arg2 ) , arg3 ) |

* **Number Functions**

1. ROUND: Rounds value to specified decimal.

ROUND(45.926) = 46

1. TRUNC: Truncates values to specified decimal.

TRUNC(45.926) = 45

1. MOD: Returns remainder of division.

MOD(1600,300)=100

**i.e:**

**SELECT ROUND(54.3434), ROUND(344.34) FROM DUAL;**

**SELECT ROUND(45.456,2), ROUND(45.999, 2 ) FROM DUAL;** (We are saying that we need only two digits after the decimal)

**SELECT TRUNC(45.999,2), TRUNC(45.123,2) FROM DUAL;**

**SELECT TRUNC(45.999,2), TRUNC(45.123,2)**

**FROM DUAL**

**/** -forward slash is for termination only in new line

**SQL> SELECT MOD(11,2), MOD(100,2) FROM DUAL;**

Q: Display the employees who are getting even salaries?

A: **SELECT \* FROM EMP WHERE MOD(SAL,2)=0;**

Q: Employees who are getting odd salaries?

A: **SELECT \* FROM EMP WHERE MOD(SAL,2)=1;**

**Q:** Display employee name with middle character from each ename?

A: **SELECT ENAME, SUBSTR(ENAME, ROUND(LENGTH(ENAME)/2),1) FROM EMP;**

Assignment:

1. Display ‘s’ in all the names.

[ Smith’s salary is 1800]

1. Using LIKE operator find a name which has %.

**Date Functions:**

**Working with Dates:**

* Oracle Database stores dates in an interrrnal numeric format: century, year, month, day, hours, minutes, seconds.
* The default date display format is DD-MON-YY.

**SELECT ENAME, HIREDATE**

**FROM EMP**

**WHERE ENAME LIKE ‘A%’;**

**Arithmetic with Dates:**

1. Add or subtract a number to or from a date for a resultant date value.
2. Subtract two dates to find the number of days between those dates.
3. Add hours to date by dividing the number of hours by 24.

**SQL> SELECT SYSDATE FROM DUAL;**

**SYSDATE**

**---------**

**30-SEP-16**

**SQL> SELECT SYSTIMESTAMP FROM DUAL;**

**SYSTIMESTAMP**

**------------------------------------**

**30-SEP-16 03.23.25.221000 PM -07:00**

**SQL> SELECT SYSDATE-59 FROM DUAL;**

**SYSDATE-5**

**---------**

**02-AUG-16**

**SQL> SELECT SYSDATE+59 FROM DUAL;**

**SYSDATE+5**

**---------**

**28-NOV-16**

**SELECT ENAME, TRUNC((SYSDATE – HIREDATE)/365)”EXPERIENCE” FROM EMP;**

**Date Types**

|  |  |
| --- | --- |
| **Date Types** | **Description** |
| 1. VARCHAR2(size) | Variable-length character data – 4000 bytes. |
| 1. CHAR(size) | Fixed-length character data- 2000 bytes |
| 1. NUMBERS(p.s) | Variable-length data- 38 bytes. |
| 1. LONG | Variable- length character data up to 2gb- Only one per table. |
| 1. DATE | Date and time values. |
| 1. CLOB | Character data upto 4-gb |
| 1. RAW and LONG RAW | Raw binary data- 2000 bytes |
| 1. BLOB | Binary data upto 4- gb |
| 1. BFILE | Binary data stored in an external file: upto 4-gb. |
| 1. - ROWID | A 64 base number system.  Representing the unique address of a row in the table. |

The difference between Varchar and char is that char is fixed in size, where as Varchar may differ in length according to the character given in the Varchar. For ex: If we reserve 4 bytes in Varchar and give a name of 3 bytes…it will only reserve 3 bytes, whereas in Char it will reserve 4 bytes.

Number(4)=1111

Number(P,S)---(number of digits , decimal)

i.e: Number(7,3)---it can have 7 digits and 3 is the number after decimal and it should be less than or equal to the first arg.

Date data type has a default format as DD-MON-YY. We can change it.

LONG- is a variable length char data.

**Conversion Functions:**

Data type conversion

Explicit data type conversion

Implicit data type conversion

**General Funcctions (Special Function):**

These function work with any data type and pertain to using nulls.

NVL(expr1, expr2)

NVL(ezpr2, expr2, expr3)

**NVL Function**

Converts a null an axtual value

Data types that can be used are Date, Character and Numbers.

Data types must match.

NVL(comm,0)

NVL(hireDate,’01-JAN-97’)

NVL(JOB,’LOOKING FOR JOB’)

**SELECT ENAME, SAL, COMM, SAL+NVL(COMM,0) FROM EMP;**

**NVL2 Function**

Converts not null and null to an actual value.

1. Data types that can be used are date, character and Numvers.
2. Data types may or may not match.

NVL2(COMM,0,1)

NVL2(COMM,’Gets comm’,’no comm’)

Returns values must have same data types, but they can differ from comparison data type.

NVL(arg1, arg2, arg3)

* If the 1st value is not null then it gives the 2nd value.
* If the 1st value is null it return the 3rd value.
* First argument can be of any data type, but arg2 and arg3 should be number data types.

**SELECT ENAME, SAL ,COMM, SAL+NVL2(COMM,COMM,0) FROM EMP;**

**SELECT ENAME, SAL, COMM, NVL2(COMM,SAL+COMM,SAL) FROM EMP;**

**Q:** Display a commission of rupees 150 for those rows which have null under the commission column?

**A:** **SELECT ENAME, SAL, COMM, NVL2(COMM, COMM, 150) FROM EMP;**

**SELECT ENAME, SAL, COMM, NVL(COMM, 150) FROM EMP;**

* **Multirow Function/Group Functions**

What are group functions?

* Group functions operate on sets of rows to give one Result per group.

1. AVG 4. MAX
2. COUNT 5.MIN
3. SUM

**MIN and MAX:**

**SELECT MIN(SAL), MAX(SAL) FROM EMP;**

**SELECT MIN(ENAM), MAX(ENAME) FROM EMP;**

**SUM:** only for columns with number data types

**SELECT SUM(SAL) FROM EMP;**

**AVG:** it neglects the null values while counting the average.

**SELECT AVG(COMM) FROM EMP;**

**COUNT:** it cancels the null value and count the actual values.

**SELECT COUNT(\*), COUNT(ENAME), COUNT(MGR), COUNT(COMM) FROM EMP;**

**Q:** Display the total number of employees in EMP table?

A: **SELECT COUNT(\*) FROM EMP;**

**Q:** Display total no. of EMP in 10th department?

A: SELECT COUNT(\*) FROM EMP WHERE DEPTNO=10;

Q: Display total no. of employee in 20th department?

**A: SELECT COUNT(\*) FROM EMP WHERE DEPTNO=20;**

Q: Display total no. of employee in 30th department?

A: **SELECT COUNT(\*) FROM EMP WHERE DEPTNO=30;**

Q: Display department wise no. of employees?

A: **SELECT DEPTNO, COUNT(\*) FROM EMP GROUP BY DEPTNO;**

**Q:** Display department wise maximum salary?

**A: SELECT DEPTNO, MAX(SAL) FROM EMP GROUP BY DEPTNO;**

* Group functions can never be used in ‘WHERE’ clause, because ‘WHERE’ clause checks one row at a time where as group functions checks all the rows at the same time
* If we want group functions to consider null values then use NVL function.
* **Creating Groups of Data:**

The GROUP BY clause syntax:

SELECT XXXXX, group\_function(column)

FROM table

[WHERE condition]

[GROUP BY XXX]

[ORDER BY column];

* Divide rows in a table into smaller groups by using the GROUP BY clause.
* All columns in the SELECT list must be in the GROUP BY clause.
* The GROUP BY column does not have to be in the SELECT list.
* You can group by more than one column, creating groups and sub group.

NOTE: Whenever we want to select the group we use ‘HAVING’ clause.

Q: Display jobwise number of employees?

A: **SELECT JOB, COUNT(\*) FROM EMP GROUP BY JOB;**

**Q:** Display jobwise minimum salary, maximum salary and average salary?

A: **SELECT JOB, MIN(SAL), MAX(SAL), AVG(SAL) FROM EMP GROUP BY JOB;**

**Q:** Display department wise number of employees?

**A: SELECT DEPTNO, COUNT(\*) FROM EMP GROUP BY DEPTNO;**

**Q:** Display department wise number of employee with those department which have more than four employees working in it?

A: **SELECT DEPTNO, COUNT(\*) FROM EMP GROUP BY DEPTNO HAVING COUNT(\*) > 4;**

Q: Display those jobs which have average salary greater than 1500?

A: SELECT JOB, AVG(SAL) FROM EMP GROUP BY JOB HAVING AVG(SAL)> 1500;

**Using ‘**GROUP’ **functions with ‘HAVING’**

You cannot use the WHERE clause to restrict/select groups.

You ust the HAVING clause to restrict/select groups.

You cannot use group functions in WHERE clause.

**Excluding Group Results: The HAVING clause**

Use the HAVING clause to restrict groups:

1. From clause.
2. Where clause restrict/ selects rows.
3. Rows are grouped (GROUOP BY)
4. Groups matching the HAVING clause are selected.
5. Order by
6. SELECT clause.

**Summary**

* Use the group function COUNT, MAX, MIN, AVG.
* Write queries that use the GROUP BY clause.
* Write queries that use the HAVING clause.

SELECT column, group\_funtion(column)

FROM table

[WHERE condition]

[GROUP BY group\_by\_expression]

[ORDER BY column];

Q: Display employee names with first and last character in the UPPER case and the middle character in lower case?

A: **SELECT CONCATE( INITCAP(SUBSTR(ENAME, 1, LENGTH(ENAME)-1)), UPPER(SUBSTR(ENAME, -1)))) FROM EMP;**

Q: Display job wise maximum salary for all the managers and salesman?

A: **SELECT JOB, MAX(SAL) FROM EMP WHERE JOB IN(‘MANAGER’,’SALESMAN’) GROUP BY JOB;**

**Q:** Display department wise minimum salary for that department which have more than four employees?

A: **SELECT DEPTNO, MIN(SAL) FROM EMP GROUP BY DEPTNO HAVING COUNT(\*) > 4;**

**Q:** Display salary wise no. of employees for those salaries which are duplicated only once?

A: SELECT SAL , COUNT(\*) FROM EMP GROUP BY SAL HAVING COUNT(\*)=2;

Q: Display job wise number of employees for those jobs which have maximum salary more than 3000?

A: **SELECT JOB,MAX(SAL), COUNT(\*) FROM EMP GROUP BY JOB HAVING MAX(SAL) > 3000;**

**Assignment [Function]**

1. Display all the employees whose job has string ‘MAN’.
2. Display all the employees whose name has ‘L’ as the third character.
3. Replace third character with ‘\*’ in ename column.
4. Display all the employees who has joined in year 81.
5. Display all the employees who has joined in FFB.
6. Display all the names which has at least 2L’s in it.
7. Display number of L’s in each name.

**Assignment [Grouping]**

1. Display job wise minimum salary.
2. Display the department that are having more than 3 employees under it.
3. Display job-wise average salaries for the employees whose employee number is not form 7788 to 7790.
4. Display dept wise total salaries for all the Managers and Analyst, only if the average salaries for the same is greater than or equal to 3000.

Q: Display those employees who have joined on Sunday?

A:

Q: Display those employees who have in a month on april?

A:

Q: Display maximum salary from emp table?

A: **SELECT MAX(SAL) FROM EMP;**

Q: Display the employee earning maximum salary?

A: **SELECT \* FROM EMP WHERE SAL = (SELECT MAX(SAL) FROM EMP);**

Q: Display the employee whose designation same as that of alex designation?

A: **SELECT \* FROM EMP WHERE JOB = (SELECT JOB FROM EMP WHERE ENAME = 'ALLEN');**

**Subqueries:**

* Query which is inside is sub query.
* Query inside query is Nested query
* Whenever we have unknown valueswe use Sub Queries.
* Sub queries are used in WHERE clause.

Subquery Syntax:

SELECT select list

FROM table

WHERE expr operator (SELECT select list FROM table);

Q: Display the employees working in same department as that of millers department as that of millers department?

A: **SELECT \* FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM EMP WHERE ENAME = 'MILLER');**

**Q: D**isplay the employeees working under king? Or Display the subordinates of king?

A: **SELECT \* FROM EMP WHERE MGR=(SELECT EMPNO FROM EMP WHERE ENAME = 'KING');**

**Q**: Display the employee who have joined after Blake?

A: **SELECT \* FROM EMP WHERE HIREDATE>(SELECT HIREDATE FROM EMP WHERE ENAME = 'BLAKE');**

**Q:** Display the employee woriking in research department?

A: **SELECT \* FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM DEPT WHERE DNAME = 'RESEARCH');**

Q: Display the employee working in chichago?

A: **SELECT \* FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM DEPT WHERE LOC='CHICAGO');**

Q: Display the employee who are working in dallas?

A: **SELECT \* FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM DEPT WHERE LOC='DALLAS');**

**Q:** Display the employee working in research or accounting department?

A: **SELECT \* FROM EMP WHERE DEPTNO IN(SELECT DEPTNO FROM DEPT WHERE DNAME IN ('RESEARCH','ACCOUNTING'));**

**Q:** Display the employees whose designation is same as that of scotts designation and salary more than millers salary?

A: **SQL> SELECT \* FROM EMP WHERE JOB =(SLECT JOB FROM EMP WHERE ENAME = 'SCOTT') AND SAL > (SELECT \* FROM EMP WHERE SAL > (SELECT SAL FROM EMP WHERE ENAME = 'MILLER');**

**Q: Di**splay Scotts manager details?

A: SELECT \* FROM EMP WHERE EMPNO=(SELECT MGR FROM EMP WHERE ENAME = 'SCOTT');

Q: Display scotts mangers information?

A**: SELECT \* FROM EMP WHERE EMPNO=(SELECT MGR FROM EMP WHERE ENAME=(SELECT MGR FROM EMP WHERE ENAME = 'SCOTT'));//WRONG**

Q: Display the employee who are earning first maximum salary?

A: **SQL> SELECT \* FROM EMP**

**2 WHERE SAL = (SELECT MAX(SAL) FROM EMP);**

**Q:** Display the employees whosE salary is more than the average salary of all the employees working in research departmetn?

A: SELECT \* FROM EMP

WHERE SAL > (SELECT AVG(SAL)

FROM EMP

WHERE DEPTNO = (SELECT DEPTNO FROM DEPT WHERE DNAME=’RESEARCH’));

Q: Display second maximum salary?

**A: SELECT \* FROM EMP WHERE SAL = (SELECT MAX(SAL) FROM EMP WHERE SAL <(SELECT MAX(SAL) FROM EMP));**

**Q:** Display third maximum salary?

A:

Q: Display 7th maximum salary?

A**: SELECT \* FROM EMP X WHERE 7=(SELECT COUNT(DISTINCT SAL) FROM EMP WHERE X.SAL <= SAL);**

**Q:** Display the employees whose salary is greater than allen’s manager salaray?

A:

**Table Alias**

**SELECT X.\*, X.ENAME FROM EMP X;**

The syntax varies when we say \* and table\_name .\*.

**SELECT X.ENAME, X.SAL, X.\* FROM EMP X;**

**SELECT X.\*,EMP.ENAME FROM EMP X; (ERROR)**

Wheneve we are giving a table alias then only the alias name should be used not the actual tabel name.

**Q: Display the employee whose salary is greater than jones manager’s salary?**

**Q: Display the employees whose salary is greater than there own manager salary?**

**A:** **SELECT \* FROM EMP X WHERE X.SAL > (SELECT SAL FROM EMP WHERE X.MGR=EMPNO);**

**Display the employees who have joined after allen’s manager?**

**Display the employees who are senior to allen’s manager?**

**Display the employees who are senior to their manager?**

**Assignment [Sub-Queries]**

1. Display all the employees who are earning more than all the managers.
2. Display all the employees who are earning more than any of the managrs.
3. Select employee number, job & salaries of all the Analyst who are earning more than any of the managers.
4. Select all the employees who work in Dalllas.
5. Select dept name & location of all the employees working for CLARK.
6. Select all the managers and clerks who works in Accounting and Marketing dept.
7. Display all the managers and clerks who works in Accounting and Marketing dept.
8. Display all the Salesmans who are not located in Dallas.
9. Select all the employees who are earning samee as SMITH.
10. Display all the employees who are getting more than the average salaries of all employees.
11. Display ALLEN’s saaal.
12. Display KING’s deptno
13. Display deptno of ‘CHICAGO’.

Q: Display the employees who is 6th maximum salary?

A: **SELECT \* FROM EMP X WHERE 6 = (SELECT COUNT(DISTINCT SAL) FROM EMP WHERE X.SAL<=SAL);**

**Q**: Display the employees who are earning maximum salary in there own department?

A: SELECT \* FROM EMP X WHERE X.SAL = (SELECT MAX(SAL) FROM EMP WHERE X.DEPTNO = DEPTNO);

Q: Display the senior most employee from each department?

A: **SELECT \* FROM EMP X WHERE X.HIREDATE = (SELECT MIN(HIREDATE) FROM EMP WHERE X.DEPTNO = DEPTNO);**

**Q:** Display the employees whose department is same as that of there managers department?

A: **SELECT \***

**FROM EMP X**

**WHERE X.DEPTNO = (SELECT DEPTNO**

**FROM EMP**

**WHERE X.MGR=EMPNO);**

Q: Display the employee who do not belong to the same department as that of there manager’s department?

A: **SELECT \***

**FROM EMP X**

**WHERE X.DEPTNO != (SELECT DEPTNO**

**FROM EMP**

**WHERE X.MGR=EMPNO);**

**Assignments on Correlated Sub queries:**

1. Write a query to get 4th max salary from emp table.
2. Write a query to get 2nd and 6th max sal.
3. Write a query to get first 3 max salaries from emp table.
4. Write a query to get second least saal from emp table.
5. Write a query to get least 3 salaries from emp table.
6. Display all the employees whose salary is greater than their managers salary .
7. Display all the employees whose dept is not samw as their managers.
8. Display all the employees who get lowest salary in their dept.

Date: 25,oct,16

**CODD’S 12 Rules**

Any system which claims to be a relational database management system must satisfy following rules.

* **Rule 0**: The system must qualify as relational, as a database, and as management system.

For a system to qualify as a relational DBMS(RDBMS), it must use its relational facilities (exclusively) to manage the database.

* **Rule 1**: information rule
* **Rule 2**: Guaranteed Access
* **Rule 3**: Systematic Representation of Null values
* **Rule 4**: Data Dictionary
* **Rule 5**: Talks about Sub-Languages
* **Rule 6**: View Updating
* **Rule 7**: High level insert, update & delete
* **Rule** **8**: Physical Independence
* **Rule 9**: Logical Independence
* **Rule 10**: Integrity Independence
* **Rule 11**: Distribution Independence
* **Rule 12**: No Subversion

//**HIGH LEVEL INSERT**

INSERT ALL

INTO DEPTCPY VALUES (2O,’TESTING’,’HYDERABAD’)

INTO DEPTCPY VALUES (30,HR,CHENNAI)

INTO DEPTCPY VALUES (40,SALES,’BIHAR’)

SELECT \* FROM DUAL;

**DATA TRANSACTION LANGUAGE**

SAVEPOINT A;

DELETE EMP123

WHERE ENAME = ‘KING’;

SELECT COUNT(\*) FROM EMP123;

SAVEPOINT B

DELETE EMP123

WHERE ENAME LIKE ‘A%’;

SAVEPOINT C;

DELETE EMP1233

WHERE DEPTNO IN(20,30);

SELECT COUNT(\*) FROM EMP123;

SAVEPOINT D;

DELETE EMP123;

SELECT COUNT(\*) FROM EMP123;

ROLLBACK TO D;

SELECT COUNT(\*) FROM EMP123;

ROLLBACK TO SAVEPOINT B;

SELECT COUNT(\*) FROM EMP123;

COMMIT;

ROLLBACK TO A;

//ERROR: at Line 1: save point ‘A’ never established

**PL/SQL**

* Procedural Language using Structured Query Language and uses to execute multiple instructions inside the Database.

**About PL/SQL:**

* PL/SQL is the procedural extension to sql with design features of programming languages.
* Data manipulation and query statements of SQL are included within procedural units of code.

Procedural Statement executer

PL/SQL

**PL/SQL**

PL/SQ

BLOCK

SQL STATEMENTS

PL/SQL Engine

**PL/SQL Block Structure**

DECLARE (optional)

Variables, cursors, user-defined exceptions

BEGIN (mandatory)

* SQL statements
* PL/SQL statements

EXCEPTION (optional)

Actions to perform when errors occur

END: (Mandatory)

DECLARE

BEGIN

EXCEPTION

**Defining our own errors for problematic situation and displaying them is called as user- defined Exceptions and will be used inside a Block.**

Example of function:

**CREATE PROCEDURE MYFIRSTPROC AS**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(‘HELLO CLASS!!!);**

**END;**

**/**

**:SET SERVEROUTPUT ON** //set the server on

**:EXECUTE MYFIRSTPROC;** // execute the query

**DECLARE**

**Employee varchar2(20);**

**Jobs varchar2(20);**

**BEGIN**

**Select ename into employee from emp where empno = 7802;**

**Select job into jobs from emp where empno = 7802;**

**DBMS\_OUTPUT.PUT\_LINE(employee||’is a’||jobs);**

**END;**

**/**

Difference between a Stored Procedure and Functions:

**Function:**

1. Should return at least one output.
2. Passed and compiled at runtime.
3. Cannot affect the state of database.
4. Can be invoked from SQL statement. Eg: SELECT.
5. Functions are mainly used to complete values.

**Procedures**

1. Doesn’t need to return values but can return value.
2. Stored as a pseudo-code in database, i.e., compiled form.
3. Can affect the state of database using commit etc.
4. Cannot be invoked from sql statement. Eg: SELECT .
5. Procedures are mainly used to process the tasks.

Triggers:

* Triggers provide a way of executing PL/SQL code on the occurrence of specific database events.
* Example:

:**CREATE TABLE ITEMS(ID NUMBER(1) PRIMARY KEY,**

**NAME VARCHAR2(10) NOT NULL,**

**PRICE NUMBER(5) NOT NULL);**

**:INSERT ALL**

**INTO ITEMS VALUES(1,’TV’,1000)**

**INTO ITEMS VALUES(2,’AC’,2000)**

**INTO ITEMS VALUES(3,’LAPTOP’4000)**

**SELECT \* FROM DUAL;**

**:SELECT \* FROM ITEMS;**

**:CREATE TABLE UPDATED\_ITEMS(U\_ID NUMBER(1) PRIMARY KEY)**

**ID NUMBER(1) REFERENCES ITEMS(ID),**

**NAME VARCHAR2(10) NOT NULL,**

**OLD\_PRICE NUMBER(5) NOT NULL,**

**NEW\_PRICE NUMBER(5) NOT NULL**

**U\_DATE DATE);**

:**CREATE SEQUENCE UPDATED\_ITEMS\_SEQ;**

**:SELECT \* FROM UPDATED\_ITEMS;**

**:CREATE TRIGGER ABCD**

**AFTER UPDATE OF PRICE ON ITEMS FOR EACH ROW**

**BEGIN**

**INSERT INTO UPDATED\_ITEMS VALUES(UPDATED\_ITEMS\_SEQ.NEXTVAL,**

**:NEW.ID,:NEW.NAME,:OLD.PRICE,:NEW.PRICE, SYSDATE);**

**END;**

**/**

**:SELECT \* FROM ITEMS;**

**:UPDATE ITEMS**

**SET PRICE = 15OO**

**WHERE ID = 1;**

**Normalization:**

Normalization is a process of efficiently organizing data in a Database.

There are two goals of the normalization process: elimination redundant data (for example, storing the same data in more than one table ) and example , storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table).

Both of these are worthy goals as they reduce the amount of space a Database consumes and ensures that Data is logically stored.

1. **First normal form(1NF)**:
2. A atomic values(Atomicity)
3. No Duplicates Columns
4. Have a Primary Key Column
5. **Second normal form: (2NF)**
6. If a table can be divided than create multiple table
7. Create foreign key between divided tables
8. **Third normal form(3NF)**
9. NO transitive functional dependency or all the columns in a table should be dependent directly on the primary key column.
10. **Forth Normal Form (4NF)**
11. No multi value dependency, that is only one primary key column per table.
12. **Fifth Normal Form(PJNF –** Project Join Normal Form**) or (5NF)**
13. All the joins condition should be return using candidate Keys.
14. **3.5 NF(BCNF) –** Boyce-Codd Normal Form

**All**  the columns in a Table should be candidate keys.

**Indexes:**

Indexes provide faster access to rows. Indexes can be created on any column. Usualy search on a column is faster if an index exists on that colum.

Syntax:

CREATE INDEX index\_name

ON table\_name(column\_name);

**CREATE INDEX ABC**

**ON EMP(ENAME, SAL);**

**DROPING INDEX ABC;**

**Views:**

Views are a way to store statements inside a Satabase

Ex:

: Create view xyz as select ename, deptno from emp;

-if getting an error, then:

: grant view to scott

:/

?select \* from xyz;

Details of views;

Select view\_name,text from user\_views;

SELECT \* FROM XYZ;

Droping a view:

: drop view xyz;

CREATE VIEW VIEW\_ABC AS SELECT ENAME, DNAME

FROM EMP, DEPT

WHERE EMP.DEPTNO = DEPT.DEPTNO;

:CONNECT SYSTEM/TIGER

:SHOW USER

:GRANT VIEW TO SCOTT;

:DROP VIEW VIEW\_ABC;

**SYNONYMS:**

Synonyms are alternate names for tables, view or any other Object in the database. They provide both data independence and location transparency.

Syntax:

CREATE SYNONYM synnm1 for emp;

If error then: grant synonym to scott;

:/

Drop synonym: drop synonym synnm1;

**:CREATE SYNONYM SYN\_ABC FOR EMP;**

**:SELECT \* FROM SYN\_ABC;**

**Database Limits**

* Row in a table is unlimited to the extend of hard disk size.
* 1000 columns per table.
* IN operator can take upto 1000 values.
* Subqueries: unlimited in from clause and 255 subqueries in where clause.
* Primary key columns – 32 in Oracle, 16 in sql Serverl

Ex. Of subquery in FROM clause:

SELECT \* FROM (SELECT ename, deptno from emp);

**SELECT \* FROM (SELECT ENAME, SAL, JOB FROM EMP);**

**DATA CONTROL LANGUAGE**

SELECT \* FROM TAB;

:CONNECT HR/TIGER

:SELECT \* FROM TAB;

: GRANT SELECT ON EMP TO HR;

:CONNECT HR/TIGER

:SHOW USER

:SELECT \* FROM SCOTT.EMP;

:REVOKE SELECT ON EMP FROM HR;

:CONNECT HR/TIGER;

:SELECT \* FROM SCOTT.EMP;

**CONDITIONAL EXPRESSIONS:**

* Provide the use of IF-THEN-ELSE logic within SQL statements.
* Use two methods
* CASE expression
* DECODE expression

**SELECT ENAME, SAL, COMM,**

**CASE ENAME WHEN ‘KING’ THEN 100**

**WHEN ‘MILLER’ THEN 200**

**WHEN ‘SCOTT’ THEN 300**

**ELSE 123 END BONUS**

**FROM EMP;**

**DECODE:**

**SELECT ENAME, SAL, COMM,**

**DECODE(JOB,’MANAGER’,3000,’ANALYST’,2000,’CLERK’,1000,1600) BONUS**

**FROM EMP;**

**ROWNUM and ROWID**

* ROWID stores address of every row of a table
* ROWNUM is dynamically generated.
* ROWNUM is the order in which rows are selected from the database.

**SELECT EMP.\*,ROWNUM,ROWID**

**FROM EMP;**

**SELECT ENAME,SAL,ROWNUM,ROWID**

**FROM EMP**

**WHERE ENAME LIKE ‘A%’;**

**Di**splay first entry from emp table?

Select \* from emp where rowed = (select min(rowid) from emp;

Display the last entry from emp table?

Select \* from emp where rowed =(select max(rowid) from emp;

Display first and second entry from emp table?

Select \* from emp where rownum<=2;

Display second entry from emp?

Select \* from emp x where 2 = (select count(rowid) from emp where x.rowid>=rowid);

Display first two and last two entries from emp table?

Select \* from emp where rownum <= 2

Or rowid = (select max(rowid) from emp)

Or rowid = (selec max(rowid) from emp

Where rowid<(select max(rowid) from emp;

Display Nth maximum salary?

Select \* from emp x where &nthmaxsal = (select count(distinct sal from emp where x.sal <=sal);