Oracle SQL

Purpose of Database

- To store the data parmanently
- Data Manipulation---insert,delete,update or change or modify,sorting,filtering

What is data (It is just values of different datatypes)

- 100 (Integer)
- Jaysukh Patel (Text or String)
- Ahmedabad (Text or String)
- 75.67 (Float or Real)
- True or False or Yes or No (Boolean)

What is Information (Meaningfull Data)

- Rollno is 100
- Name is Jaysukh Patel
- Address is Ahmedabad
- Percentage is 75.67
- Married/Unmarried is True or False

Database History (File Management System)

- Text file
- CSV (Comma Seperated Value) File
- TSV (Tab Seperated Values) File

For Example Student.txt file or Student.csv file

Rollno,Name,Address
1,jay,naranpura
2,kiran,maninagar

For Example Student.tsv file

Rollno	Name	Address
1	jay	maninagar
2	kiran	isanpur

DBMS--Database Managment System

- Data insert,data delete,update,sorting,filtering operations

RDBMS--Relational DBMS(Schema Based(Table) or Skeleten Based Database)

- Microsoft Excel(Microsoft)--Table Format(Row and Column Format)
- Microsoft Access(Microsoft)--Table
- Oracle(Oracle)—Table
- MS SQL Server(Microsoft)--Table
- My SQL(Oracle)--Table
- PostgreSQL--Table

SQL (Structured Query Language)

- It is a common language to communicate with any relational database.
- **70% to 80% syntax(SQL statements Keywords, Rules and Regulations)** are same for all the relational database.
- 20% to 30% syntax is specific to database that you can learn yourself .

List of Relational Databases(SQL Databases)

- Microsoft Access(Microsoft)--SQL
- Oracle(Oracle 8,8I,9I,10g,11g,12c,19c)—SQL
- MS SQL Server(Microsoft)--Transact SQL (T SQL)
- My SQL(Oracle)--SQL
- PostgreSQL--SQL

NOSQL(Not Only SQL) Database

- Not Only SQL, meaning that NoSQL databases have the specificity of not being relational because they can store data in an unstructured format.
- NoSQL databases use a dynamic schema to query data. Also, some NoSQL databases use SQL-like syntax for document manipulation.

Advantages

- **High Scalability:**It can handle increasing demand by adding more servers(machines) to the infrastructure.
- **Global High Availability:**They can run continuously without interruption of service. Can access the same data simultaneously through different machines from different geographical zones because the database is shared globally.
- **Efficient Big Data Manangement:**Storage capacity of large amount of unstructured data make them good fits for bigdata.
- **Multimode Data:**NoSQL databases offer more flexibility than traditional SQL databases because they can store structured (e.g. data captured from sensors), unstructured (images, videos, etc.), and semi-structured (XML, JSON, etc.) data.
- **Flexibility:** NoSQL databases can rapidly adapt to changing requirements with frequent

updates and new features.

NoSQL Databases vs. SQL Databases

	SQL Databases	NoSQL Databases
Language	SQL databases use structured query languages to perform operations, requiring the use of predefined schema to better interact with the data.	On the other hand, NoSQL databases use a dynamic schema to query data. Also, some NoSQL databases use SQL-like syntax for document manipulation.
Data Schema	SQL databases have a predefined and fixed format, which cannot be changed for new data.	NoSQL databases are more flexible. This flexibility means that records in the databases can be created without having a predefined structure, and each record has its own structure.
Scalability	SQL databases are only vertically scalable, meaning that a single machine needs to increase CPU, RAM, SSD, at a certain level to meet the demand.	NoSQL databases are horizontally scalable, meaning that additional machines are added to the existing infrastructure to satisfy the storage demand.
Big Data Support	The vertical scaling makes it difficult for SQL databases to store very big data (petabytes).	The horizontal scaling and dynamic data schema make NoSQL suitable for big data. Also, NoSQL databases were developed by top internet companies (Amazon, Google, Yahoo, etc.) to face the challenges of the rapidly increasing amount of data.
Properties	SQL databases use the ACID (Atomicity, Consistency, Isolation, Durability) property.	NoSQL databases, on the other hand, use the CAP (Consistency, Availability, Partition Tolerance) property.

List of NOSQL(Not Only SQL) Databases

- MongoDB
- Cassandra
- Elasticsearch

- Neo4J
- HBase
- CouchDB
- OrientDB

When should NoSQL databases be used?

In this fast-growing and competitive environment, industries need to collect as much data as possible to satisfy their business goals. Collecting data is one thing, but storing them in the right infrastructure is another challenge. The difficulty comes because data can be of different types such as images, videos, text, and sounds. Using relational databases to store these different data types is not always a smart move. However, the question remains:When to use NoSQL instead of SQL?

You should consider using NoSQL when you are in the following scenario:

- Constant changing of data: when you do not know how your system or applications will grow in the future, meaning that you might want to add new data types, new functions, etc.
- A lot of data: when your business is dealing with huge data that might grow over time.
- No consistency: when data consistency and 100% integrity are not your priority. For example, when you develop a social media platform for your business, all the employees seeing your posts at once might not be an issue.
- Scalability and cost: NoSQL databases allow greater flexibility and can control costs as your data needs change.

Database Treminology

- **Database(Schema)**-First you have to create database or schema(username and password)
- **Tables(Entity)-**In database you can create any no of tables to store data.
- **Columns(Attributes)**-In table no of columns are there to store different information.
- **Records or Rows or Tuples-**you can store no of records in to the table.
- **ER model or ER Diagram**- Enity Relationship Diargram
- **Primary Key-**It is used to make all the records of the table unique.
- Duplicates and null is not allowed in primary key column.
- You can not put more than one primary key into the table.
- **Foregin key-**Foreign key column actually refers some primary column.
- In Foreign key column Duplicate is allowed,null value is also allowed.You can enter any value that is available in the primary key column.
- A table can have nore than one foreign keys
- Primary and foreign key may be in the same table or in different table
- Primary key and foreignkey column name may be same or different but datatatype must match

Purpose of Constraints

- it is used to prevent invalid data entry into the table.

SQL Constraints

- Primary Key (No duplicate and No NULL values)
- Foreign Key (Duplicate or NULL is allowed)
- Check (Value is allowed if it satisfies the check condition)
- Unique (No duplicate but NULL values are allowed)
- Not Null(Value is mandatory means NULL is not allowed)

SQL Statements

- **DQL—**Data Query Language (Select)
- **DML--**Data Manipulation Language (insert,update,delete,merge)
- **DDL--**Data Definition Language (create,alter,drop,rename,truncate,comment)
- **DCL--** Data Control Language (grant,revoke)
- **TCL--**Transaction Control Language (commit,rollback,savepoint)

Development Environment for Oracle SQL

- **SQL Plus**—Command based
- **SQL Developer**—GUI(Graphical User Interface) based

Sample HR Schema or Database

Tables

1)Employees

Employee_id(PK)
Department_id(FK) references D**epartemnt_id(PK) from Departments table**
Manager_id(FK) references **Employee_id(PK) from Employees table**

2)Departments

Department_id(PK)
Manager_id(FK) references **Employee_id(PK) from Employees table**
Location_id(FK) references **Location_id(PK) from Locations table**

3)Locations

Location_id(PK)
Country_id(FK) references **Country_id(PK) from Countries table**

4)Countries

Country_id(PK)

Region_id(FK) references **Region_id(PK) from Regions table**

5)Regions
Region_id(PK)

6)Jobs

7)Job_history

8)job_grades

Oracle SQL commands

spool command

It is a command to make spool file. You can record your all sql commands and output into a text file.

Syntax
spool path for directory or folder\filename.txt;

For Example
spool D:\sql\vedant\file1.txt;

show lines—It will display you the default line size(80) of SQL Plus environment.

set lines 300—It will set the linesize to 300.

show pages--It will display you the default page size(14) of SQL Plus environment.

set pages 20---It will set the pagesize to 20.

save d:\sql\vedant\q1.sql---It will save your recent statement written on SQL Plus environment.

If you want to execute this statement from this q1.sql, then you can use following command.

@ **d:\sql\vedant\q1.sql—**It will execute the command from this file.

ed----It will open recent statement into an editor like notepad so you can edit it.
After editing your statement you have to save this statement and close the editor and place forward slash(/) on SQL prompt to execute this statement.

DQL(Data Query Language)

Select Statement

To extract data from database

Syntax

select * from tablename;
select col1,col2,col3 from tablename;

select, from, tablename,column name are not case sensitive.

Example

select all columns

select * from departments;

select specific column

select department_id,department_name from departments;

Arithmetic Expression

Add----------(+)-----Date and Number datatype
Subtract-------(-)-----Date and Number datatype
Multiply------ (*)-----Number datatype
Divide---------(/)------Number datatype

Example

select last_name,salary,salary+1000 from employees;

Rules of Precedence

Multiplication and Division occurs before addition and subtraction.
Operators of the same priority are evaluated from left to right.

Example

select last_name,12*salary+100, 12*(salary+100) from employees;

NULL value

NULL is a value that is unavailable, unassigned,unknown or inapplicable.
NULL is not the same as zero or blank space.

Any aithmetic expression using NULL values results into NULL.

Example

select last_name,salary,commission_pct from employees;

select last_name,12*salary*commission_pct from employees;

Column Alias

Used for column heading.
Usefull in calculation.
Immediately follows the column name(as keyword optional).
Requres double quotation marks if it contains spaces or special characters, or if it is case sesitive.

Example

select last_name as name,commission_pct comm from employees;

select last_name "Name",salary*12 "Annual Salary" from employees;

Concatenation Operator

Concat more than one columns or character string.
Represented by two vertical bar (||).
Create resultant column as a character expression.
Columnname || NULL will give Columnname.

Example

select last_name || job_id as "Employees" from employees;

Literal values

Date and Character values must be enclosed within single quotation mark.

Example

select last_name || ' is a ' || job_id as "Employees Details" from employees;

select last_name || ' : 1 Month salary = ' || salary as “Monthly Salary” from employees;

Alternative quote(q) operator

You can user any delimiter({ },[ ],< >,( ),$,@,#)

Example

select department_name ||' Department''s Manager_id ' || manager_id as "Departments and Manager" from departments;

select department_name ||q'(Department's Manager_id)' || manager_id as "Departments and Manager" from departments;

Distinct keyword

To suppress the duplicate rows and to tech only unique rows.

Example

select department_id from employees;
select distinct department_id from employees;

select distinct department_id , job_id from employees;
(It will display every distinct combination of column)

Describe command

To display the structure of a table.

desc tablename or describe tablename;

It display column name, datatype its size and not null

Restricting data

Syntax

select * from tablename where condition;

select col1,col2,col3 from tablename where condition;

Column alias can not be used in where clause.

=	Equal to For number, For date(use single quotation and format sesitive(DD-MON-RR)) For character values(use single quotation and case sesitive)
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>, !=, ^=	Not equal to
between	Between two values(inclusive)
in(set)	Match any of a list of values
like	Match a character pattern
is null	To find the null values.

wild card characters for pattern matching in like

%	Denotes zero or more characters.
_	Denotes one characters.

Example

select first_name from employees where first_name like 'S%';

select last_name,hire_date from employees where hire_date like '%05';

select last_name from employees where last_name like '_o%';

You can use escape identifier to search for the actual % or _ symbols.

select last_name from employees where last_name like '%_%' escape '\';

select last_name from employees where last_name like '%\%%' escape '\';

Logical Operators

And	It returns true if all the conditions are true otherwise false.
Or	It returns true if any one condition is true otherwise false.
Not	It returns true if condition is false.

It returns false if condition is true.

Rules of Precedence

1 Arithmetic operators
2 Concatenation operators
3 Comparision conditions
4 Is null ,is not null, like, not like,in, not in
5 Between, not between
6 Not equal to
7 Not logical operator
8 And logical operator
9 Or logical operator

Example

select last_name,department_id,salary from employees where department_id=60 or department_id=80 and salary>10000;

select last_name,department_id,salary from employees where (department_id=60 or department_id=80) and salary>10000;

Sorting rows by using ORDER BY clause

Syntax

select * from tablename where condition order by (column name or column position or alias) asc or desc

ASC Ascending order(Default)
DESC Descending order

You can specify multiple expressions in ORDER BY clause.

Default sort order in ascending

Numeric values are displayed with lowest values first(1 to 999)
Date values are displayed with earliest values first(01-jan-92 before 01-jan-95)
Character values are displayed in alphabetical order('A' first and 'Z' last)
Null values are displayed last for ascending sequence and first for descending sequences.
You can also sort by column that is not in the select list but result will not be meaningfull.
Use the keywords NULLS FIRST to display NULL values first

Use the keywords NULLS LAST to display NULL values last

SQL row limiting clause

You can use this clause to limit the rows that are return by the query.
It is used to implement the Top N reporting or analysis.
You can specify no of rows or percentage of rows to return.
Top N queries sort their resultset and then return only the first N rows.
You must specify ORDER BY clause with WITH TIES clause

Syntax

select * from tablename where condition order by columnname offset (row | rows) fetch (first | next) row_count | percent (row | rows) (only | with ties)

OFFSET

To specify the no of rows to skip. It must be a number. If you specify negative number offset is treated as 0. if you specify NULL or a number grater than or equal to no of rows that are returned by the query, 0 rows are returned.

FETCH

To specify no of rows or percentage of rows to return.

Example

select employee_id,first_name from employees order by employee_id fetch first 5 rows only;

select employee_id,first_name from employees order by employee_id offset 5 rows fetch next 5 rows only;

select employee_id,first_name from employees order by employee_id fetch first 5 rows with ties ;

Substitution Variables

By using substitution variable in place of exact values in where clause, you can run the same query for different values.

It can be used in **where condition, ORDER by clause, column expression, table name.**

Use single quatation mark for date and character values('&variablename' or

'&&variablename').

Single ampersend substitution variable

Syntax

&variablename

It will ask to enter value every time you execute the query.

Double ampersend substitution variable

Syntax

&&variablename

It will ask to enter value first time you execute the query. There after it will use this value.

Example

select employee_id,last_name from employees where employee_id=&emp_id;

select employee_id,job_id from employees where job_id='&job_title';

select employee_id,job_id,last_name,&column_name from employees where &condition order by &order_column;

select * from &table_name;

select employee_id,last_name,job_id from employees order by &&column_name;

Define Command

Syntax

define variable name=value;

It is used to create and assign a value to a variable.

This variable is present in the session until the user undefines it or exit the SQL Environment.

Undefine Command

Syntax

undefine variable name;

It is used to remove a variable.

Example

define employee_num=200;

select employee_id,last_name,salary from employees where employee_id=&employee_num;

select first_name,last_name,manager_id from employees where manager_id=&employee_num;

select department_name,location_id from departments where department_id=&employee_num;

undefine employee_num;

Verify command

Syntax

set verify on;

It will display you the value replaced at the place of substitution variable.

set verify off;

It will not display you the value replaced at the place of substitution variable.

SQL PLUS System variables

show all

It will display you the complete list of all system variables.

Inbuilt or Predefined SQL functions

Two types of functions

1)Single row functions

Return one result per row.

It operates on single row.

Can be nested that means you can use ouput of one func to the input of other func.

Can be used in SELECT, WHERE AND ORDER BY clause.

2)Multiple row functions or group functions

Return one result per set of rows or multiple rows or per group of rows.
It operates on multiple rows.

Categories in Single row function

Character	Accept char input and can return char or number.
Number	Accept number input and return number.
Date	Operates on date values.
Conversion	Convert a value from one datatype to another.
General	It can take any datatype and can handle NULL.

Character functions

lower(column|expression) Convert to lowercase

upper(column|expression) Convert to uppercase

initcap(column|expression) Convert the first letter of each word to uppercase

concat(column1|expression1,column2|expression2) Concate both columns or expressions

substr(column|expression , m , n) n is optional.

Rerturn substring from string starting from char position m, n characters long.
If m is negative count starts from the end of the character value.
If n is omitted all characters to the end of the string are returned.

length(column|expression) It returns the no of characters.

instr(column|expression, 'String', m, n) m and n are optional.

Return the position of char or string.
You can provide position m to start searching and the occurance of n of the string.
m and n default to 1

lpad(column|expression, n, 'String')

Returns an expression left padded to length of n characters.

rpad(column|expression, n, 'String')

Returns an expression right padded to length of n characters.

trim(leading | trailing | both, trim_character from trim_source)

Trim leading or trailing characters or both from characters string.
Trim character is the character to trim.
Trim source is the character string in which you want to trim particular character.

replace(text, serch string, replacement string)

Searches a text expression for char string and if found, replace it with replacement string

Example

select 'My name is '||upper(first_name)||' with job_id is '||lower(job_id) as “Employee Details” from employees;

select first_name,last_name,job_id from employees where lower(last_name)='higgins';

select first_name,last_name,job_id from employees where upper(last_name)='HIGGINS';

select first_name,last_name,job_id from employees where initcap(last_name)='Higgins';

select last_name,upper(concat(substr(last_name,1,8),'_US')) from employees where department_id=55;

Number Functions

round(column | Expression, n)

Round values to the n specified decimal.
If n is omitted no decimal places.
If n is negative numbers to the left of the decimal point are rounded.

round(34.936, 2) 34.94

Trunc(column | Expression, n)

Truncates values to the n specified decimal.
If n is omitted no decimal places.
If n is negative numbers to the left of the decimal point are truncated.

trunc(34.936, 2) 34.93

Ceil(column | Expression)

Returns smallest whole number equal to or greater than specified number.

Ceil(2.73) 3

Floor(column | Expression)

Returns largest whole number equal to or less than specified number.

floor(2.73) 2

Mod(m, n)

Returns remainder of m divided by n.

mod(1600,500) 100

DUAL Table

It is a public table that you can use to view results from functions and calculations.
It is owned by user SYS and can be accessed by all users.
It contains one column DUMMY and one row with value X.
You can use this table to complete the select statement.

Date Functions

Internally oracle stores any date in numeric format(century,year,month,day,hours,minutes and seconds)
The default date display fromat is DD-MON-RR(RR means last two digits of the year)

Current Year	Specified Date	RR Format	YY Format
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095
2048	27-OCT-52	1952	2052
2051	27-OCT-47	2147	2047

		If the specified two digits year is :	
		0 - 49	50 - 99
If two digits of the current year are :	0 - 49	The return date is in the current century	The return date is in the century before the current one
	50 - 99	The return date is in the century after the current one	The return date is in the current century

Sysdate function

It returns system date and time(server date and time) for the operating system on which the database resides.

current_date function

It returns current date and time in the session timezone of client or user.

sessiontimezone

It returns the value of the current session's time zone.

It is a time zone offset(+ | - TZH:TZM) or a time zone region name depending on the how user specified seesion time zone value in the most recent alter session statement.

current_timestamp

It returns current date and time from the user session.

Arithmetic operations with Dates

Operation	Result	Description
Date + Number	Date	Adds a number of days to date.
Date - Number	Date	Subtracts a number of days from a date.
Date - Date	No of Days	Subtracts one date from another.
Date + Number/24	Date	Adds a number of hours to date.

Months_between(Date1, Date2)

Number of months between two dates.
It can be positive or negative.
If date1 > date2 then positive otherwise negative.

Add_months(Date, n)

Add n number of calender months to date.
The value of n must be an integer and can be negative.

Next_day(Date, 'Char')

Finds the date of next specified day of theWeek('char').
Char may be number representing a day or string(1 , 2 or 'Mon' , 'Tuesday')

Last_day(Date)

Last day of the month.

Round(Date, 'Format')

Round date.
If format is omitted then date rounded to the nearest day.
Format my be 'month' or 'year'.

if the format is 'month' date 1-15 result in first day of the current month.
Date 16-31 result in the first day of the next month.
If the format is 'year' months 1-6 result in january 1 of the current year.
Months 7-12 result in january 1 of the next year.

Trunc(Date, 'Format')

Truncate Date.
If format is omitted then date rounded to the nearest day.
Format my be 'month' or 'year'

Two types of conversion

Implicit data type conversion

This conversion is done by oracle.
It is according to certain rules.

Varchar2 or char to number(department_id = '90')
char to number converion is succeed only if the char strring represents valid number.

Varchar2 or char to date(hire_date > '01-jan-90')
varchar2 to date conversion is succeed only if the char string represents valid date(DD-MON-RR).

Explicit data type conversion

This conversion is done by user.
It is performed by user by using data type conversion functions.

Datatype conversion functions

To_char(Number | Date , 'Format')

Convert a number or date value to varchar2.

To_date(Char | Varchar2 , 'Format')

Convert a char or varchar2 value to date.
If format is omitted the format is DD-MON-YY

TO_CHAR functtion with Date
To_char(Date , 'Format')

Format must be enclosed with single quatation marks.
It is case sensitive.
Can include any valid date fromat element.
Has an fm element to remove padded blanks or supress leading zeros.
Format model for '11-Nov-2000' is 'DD-Mon-YYYY'
It converts date from default format(DD-MON-RR) to format you specified.
The name of days and month automatically padded with blanks

Example

select last_name,hire_date,to_char(hire_date,'MM/YY') from employees;

Date format lements

YYYY	Full year in numbers
YEAR	Year spelled out in english
MM	Two digit value for the month
MONTH	Full name of the month
MON	Three letter abbreviation of the month
DY	Three letter abbreviation of the day of the week
DAY	Full name of the day of the week
DD	Numeric day of the month
HH24:MI:SS AM	14:34:32 PM
DD “of” MONTH	23 of OCTOBER
ddspth	fourteenth
AM or PM	14:34:32 PM
A.M or P.M	14:34:32 PM
HH OR HH12	12 hour format
HH24	24 hour format
MI	Minute(0 to 59)
SS	Second(0 to 59)

/ . ,	It will print as it is
“of the”	Quated string will print as it is
TH	DDTH for 4TH
SP	Spelled out number(DDSP for FOUR)
SPTH or THSP	Spelled out number(DDSPTH for FOURTH)

TO_CHAR functtion with Number

Formate elements

9	Represents a number(99999 for 12345)
0	Forces zero to be displayed(099999 for 001234)
$	Place $ sign($99999 for $12345)
. or D	For decimal point(9999.99 for 1234.00)
, or G	Thousand seperator(99,999 for 12,456)
MI	Minus signs to right(9999MI for 1234-)

Example

select salary, to_char(salary, '$99,9999.00') from employees;

If format width is smaller than width of value to be displaed then ###### will be diplayed.

TO_DATE function

It is having an fx modifier.
It specifies the exact match for the char argument and format model.
Punctuation and quoted text must exactly match.
Character arguments can not have an extra blanks. Without fx it ignores extra blanks.
Numeric data in the character argument must have the same number of digits as the corresponding element in the format.
Without fx, the number and char argument can omit leading zeros.

Example

select last_name,hire_date from employees where hire_date = to_date('May 24,2007' , 'fxMonth DD,YYYY');

To find employees hired before 1990.

select last_name,to_char(hire_date , 'DD-MON-YYYY') from employees where hire_date <'01-JAN-90' ;

General Fuctions

These functions work with any data type and can manage the NULL values.

NVL(expression1, expression2)

Converts a NULL value to an actual value.
Datatypes must match.
Exp1 is source value that may contain NULL.
Exp2 is target value for converting the NULL.
NVL(commission_pct,0)
NVL(hire_date,'01-jan-01')
NVL(job_id,'No Job')

select first_name,salary,commission_pct,(12*salary) +(12*salary*NVL(commission_pct,0)) "Annual Salary With Commission" from employees;

NVL2(expression1, expression2, expression3)

If exp1 is not NULL, it returns exp2. If exp1 is NULL it returns exp3. Exp1 can have any data type.

select first_name,salary,commission_pct,NVL2(commission_pct , 'SAL+COM' , 'SAL') from employees;

Exp1 is source value that may contain NULL.
Exp2 is the value that is return if exp1 is not NULL.
Exp3 is the value that is return if exp1 is NULL.
Exp1 can have any data type.
Exp2 and exp3 must be the same data type.

NULLIF(expression1, expression2)

Compares two expressions and returns NULL if they are equal, returns the first expression if they are not equal.

Select first_name,length(first_name),last_name,length(last_name), NULLIF(length(first_name), length(last_name)) "Result" from employees;

COALESCE(expression1, expression2.....expression n)

returns the first not NULL expression in the expression list.
All expressions must be of the same data type.

Select last_name,salary,commission_pct,COALESCE((salary+(salary*commission_pct)), salary+2000) "New Salary" from employees;

Conditional functions

It provides IF THEN ELSE logic.

Case Expression

CASE EXPRESION complies with ANSI SQL.

Syntax

```
case expr when comparision_expr1 then return_expr1
          when comparision_expr2 then return_expr2
          ------------------------------------------------
          when comparision_exprn then return_exprn
          else else_expr
end;
```

It searches for first WHEN...THEN pair for which expr is equal to comparision_expr and return return_expr. If none of the WHEN...THEN pair meets this condition returns else_expr.

Expr and comparision_expr must be of the same data type.
All the return_expr must have same data type.

```
Select last_name,job_id,salary,
       case job_id when 'IT_PROG' then 1.10*salary
                   when 'ST_CLERK' then 1.15*salary
                   when 'SA_REP' then 1.20*salary
                   else salary end  "New Salary" from employees;
```

Search Case Expression

Syntax

```
case when condition1 then return_expr1
     when condition2 then return_expr2
```

```
        ------------------------------------------------
        when conditionn then return_exprn
        else else_expr
    end;

    Select last_name,job_id,salary,
        (case when job_id='IT_PROG' then 1.10*salary
            when job_id='ST_CLERK' then 1.15*salary
            when job_id='SA_REP' then 1.20*salary
            else salary end) “New Salary” from employees;

    Select last_name,job_id,salary,
        (case when salary<5000 then 'Low'
            when  salary<10000 then 'Medium'
            when salary<20000 then 'Good'
            else 'Excellent' end) “Salary Type” from employees;
```

DECODE function

Decode function specific to oracle.

Syntax

decode(column | Expression, search1, result1,search2,result2.....searchn,resultn,default result)

If the default value is omitted NULL value is returned.

```
    Select last_name,job_id,salary,
        decode(job_id, 'IT_PROG',1.10*salary,
                    'ST_CLERK', 1.15*salary,
                    'SA_REP',1.20*salary,
                    salary) “New Salary” from employees;

    Select last_name,salary,
        decode(trunc(salary/2000,0), 0,0.00,1,0.09,2,0.20,3,0.30,4,0.40,5,0.42,6,0.44,0.45)
“Tax Rate” from employees;
```

Multirow functions or Group functions

It is used to obtain the summary information such as (sum,average,max,min) for groups of

rows.

It operates on set(It may be enitere table as one group or the table that is split in the groups) of rows to give one result per group.

AVG(distinct | all columnname)

Average value of n, ignores NULL value

COUNT(*)

It counts the total no of rows including duplicates and NULL.

COUNT(columnname)

It counts the total no of rows including duplicates but not NULL.

COUNT(distinct columnname)

It counts the total no of unique rows but not NULL .

MAX(distinct | all columnname)

Maximum value, ignores NULL

MIN(distinct | all columnname)

Minimum value, ignores NULL

SUM(distinct | all columnname)

Sum value, ignores NULL

Syntax

```
select  groupfunction1(columnname),groupfunction2(columnname).....
from tablename where condition;
```

Group function is placed after the select keyword.
You can use multiple group functions seperated by comma.
Distinct makes the function consider only nonduplicate(unique) values.
All makes it consider every value including duplicates. By default it uses all.
All group funcions ignore NULL values. So you have to use NVL,NVL2 etc functions to replace
NULL value to some actual value.
Use SUM and AVG for numeric data.
MAX and MIN can be used for numeric, text and date.

Example

select sum(salary), avg(salary), max(salary), min(salary) from employees;

select sum(salary), avg(salary), max(salary), min(salary) from employees where job_id='SA_REP';

select max(hire_date), min(hire_date) from employees;

select max(first_name), min(first_name) from employees;

select count(*) from employees where department_id=55;

select count(commission_pct) from employees where department_id=55;

select count(distinct department_id) from employees;

Group functions and NULL values

All group funcions ignore NULL values. So you have to use NVL,NVL2 etc functions to replace it.

select avg(commission_pct) from employees; //Wrong Aproach

select avg(nvl(commission_pct,0)) from employees; //Right Aproach

Grouping rows by GROUP BY Clause

You can devide your entire table into the no of smaller groups by using GROUP BY Clause.

Syntax

select columnname, groupfunction(columnname) from tablename where condition group by (columnname or group by expression) order by (columnname or expression)

If you use any colunname with group function then you have to use group by clause with that columnname otherwise It will give error.

If you use more than one colunn with group function then you have to use group by clause with that much of columns otherwise It will give error.

In short All the columns in the select list that are not group functions must be in the group by clause.

Using where clause, you can exlude rows before deviding them into groups.

Example

select department_id, sum(salary), avg(salary), min(salary), max(salary) from employees group by department_id;

Group by column does not have to be in a select list.
But select list(all the columns) must have to be in the group by clause.

select sum(salary), avg(salary), min(salary), max(salary) from employees group by department_id;

You can also use group function in ORDER BY clause.

select department_id, avg(salary) from employees group by department_id order by avg(salary);

Grouping by more than one column

Example

select department_id, job_id, sum(salary) from employees group by department_id,job_id order by job_id;

select department_id, job_id, sum(salary) from employees where department_id >40 group by department_id,job_id order by department_id;

General Erros in Group by clause

select department_id,count(first_name) from employees; //error

select department_id,job_id,count(first_name) from employees group by department_id; //error

You can not use where clause to restrict groups.
You can not use group functions in where clause.
You have to use having clause to restrict groups.

select department_id, avg(salary) from employees where avg(salary) >12000 group by department_id; //error

select department_id, avg(salary) from employees group by department_id having avg(salary) > 12000; //Right Aproach

GROUP BY with HAVING clause

You can place having clause before or after the group by clause.
It will do the following task.
Rows are grouped.
The group function is applied.
Groups matching having clause are displayed.

Syntax

select columnname, groupfunction(columnname) from tablename where condition group by (columnname or group by expression) having group condition order by (columnname or expression)

Example

select department_id, max(salary) from employees group by department_id having max(salary)>10000;

select department_id, max(salary) from employees group by department_id having avg(salary)>10000;

select job_id,sum(salary) from employees where job_id not like '%REP%' group by job_id having sum(salary)>13000 order by sum(salary);

Nesting in group functions

Group function can ne nested in depth of two functions only.
Group by cluase is mandatory when nesting group function.

Example

select max(avg(salary)) from employees group by department_id;

SQL JOINS

It is used to obtain a data from multiple table.
You can join more than one tables to view the information from more than one table.

Types of SQL JOINS compliant with SQL:1999 standard

INNER JOINS
- Natual Join with NATURAL JOIN clause**(Equijoin)**
- Join with USING clause**(Equijoin)**
- Join with ON clause**(Equijoin or NonEquijoin or self join)**

OUTER JOINS
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

CROSS JOIN

Sytanx

```
select table1.column, table2.column
from table1

NATURAL JOIN table2------------(1)
OR
JOIN table2 USING(COLUMNNAME)--------(2)
OR
JOIN table2 ON (TABLE1.COLUMNNAME=TABLE2.COLUMNNAME)-----(3)
OR
(LEFT OR RIGHT OR FULL) OUTER JOIN table2 ON (TABLE1.COLUMNNAME =
TABLE2.COLUMNNAME)----------(4,5,6)
OR
CROSS JOIN table2----------(7)
```

EQUI JOINS(INNER JOINS)

NATURAL JOIN

Joins two tables based on all the same column name and same datatype.
It select rows from two tables that have equal values in all matched columns.
If the column having same name but different data type, an error is returned.
It uses all the columns with matching names and data types to join the table.

Do not qualify a column that is used in natural join.

Example

select * from departments natural join locations;

select employee_id,department_name from employees natural join departments;

select department_id,department_name,city from departments natural join locations where department_id in(20,30,55);

2)USING Clause

If several columns have the same name but having different data type , use the using clause to specify the column to join.
It is used to match only one column when more than one column matches.
Performs an equijoin based on column you specified.

Example

select employee_id,last_name,department_id,location_id from employees join departments using(department_id);

select employee_id,last_name,department_id,location_id from employees join departments using(manager_id);

Qualifying Ambiguous column name

Use table prefixs to qualify column names that are in multiple tables.
If you qualify column name with table prefixes then it increses the speed of parsing of the statement.
Insted of full table name prefixes, use table alias like (employees e, departments d)
Shorter name keeps sql code smaller, uses less memory.
Use column alias to distinguish columns that have same name, but reside in different tables.
Do not qualify a column that is used in using clause.
The columns that are common in both the tables but not used in using clause must be prefixed with table alias.

Example

select l.city, d.department_name from locations l join departments d using(location_id) where d.location_id=1400; //Error

select l.city, d.department_name from locations l join departments d using(location_id) where location_id=1400; //Right

select first_name, d.department_name,manager_id from employees e join departments d using(department_id) where department_id=50; //Error

select first_name, d.department_name,d.manager_id from employees e join departments d using(department_id) where department_id=50; //Right

3)ON Clause

Performs an equijoin based on the condition in on clause.
Performs a Noneequijoin based on the condition in on clause.
Performs a selfjoin based on the condition in on clause.
You can also use ON clause to join columns that have different names but same data type.

Example

select employee_id,last_name,d.department_id,location_id from employees e join departments d on(e.department_id=d.department_id);

select employee_id,last_name,d.department_id,location_id,d.manager_id from employees e join departments d on(e.manager_id=d.manager_id);

select employee_id,department_name,city from employees e join departments d on(d.department_id=e.department_id) join locations l on(d.location_id=l.location_id)

select employee_id,department_name,city from employees e join departments d using(department_id) join locations l using(location_id)

select employee_id,department_name,city from employees natural join departments natural join locations;

Applying additional conditions with join

You can use **and** or **where** for ON Clause.
You can use only **where** for USING and NATURAL JOIN Clause.

Example

select employee_id,last_name,d.department_id,e.manager_id from employees e join departments d on(e.department_id=d.department_id) and e.manager_id=149 ;

OR

select employee_id,last_name,d.department_id,e.manager_id from employees e join departments d on(e.department_id=d.department_id) where e.manager_id=149 ;

select employee_id,department_name,city,manager_id from employees natural join departments natural join locations where manager_id=149;

select employee_id,department_name,city from employees e join departments d using(department_id) join locations l using(location_id) where e.manager_id=149;

SELFJOIN using ON clause

It is just joining table to itself.

Example

select w.last_name emp,m.last_name mgr from employees w join employees m on (w.manager_id=m.employee_id);

or

select w.last_name emp,m.last_name mgr from employees w join employees m on (w.employee_id=m.manager_id);

select e1.employee_id,e1.salary from employees e1 join employees e2 on (e1.salary=e2.salary) and e1.employee_id!=e2.employee_id;

NONEEQUIJOIN

It is used condition other than equality operator.

Example

select e.last_name,e.salary,j.grade_level from employees e join job_grades j on (e.salary between j.lowest_sal and j.highest_sal);

OUTER JOINS

The join of two tables returning only matched rows is called inner join.
Joining tables with NATURAL JOIN, USING and ON clause results in an inner join.
If the row does not satisfied the join condition the row does not appear in the query result.
To return the unmatched rows use the outer join.

Types of OUTER JOINS

LEFT OUTER JOIN

A join between two tables that returns the result of the inner join as well as the unmatched rows from left table is called left outer join.

Left outer join result= result of inner join + unmatched rows from left table

Example

select e.last_name,e.department_id,d.department_name from employees e left join departments d on (e.department_id=d.department_id);

RIGHT OUTER JOIN

A join between two tables that returns the result of the inner join as well as the unmatched rows from right table is called right outer join.

right outer join result= result of inner join + unmatched rows from right table

Example

select e.last_name,e.department_id,d.department_name from employees e right join departments d on (e.department_id=d.department_id);

FULL OUTER JOIN

A join between two tables that returns the result of the inner join as well as the unmatched rows from left table and the right table is called full outer join.

Full outer join result= result of inner join + unmatched rows from left table +

unmatched rows from right table

select e.last_name,e.department_id,d.department_name from employees e full join departments d on (e.department_id=d.department_id);

CROSS JOIN

It is a join of every row of one table to every row of another table.
It is called cartesian product of two table.
It generates large no of rows and result is rarely usefull.
It is usefull for some tests when you need to generate large no of rows to simulate a reasonable amount of data.

Example

select last_name,department_name from employees cross join departments;

SUBQUERY

It is used in the where clause of another sql statement to obtain values based on an unknown conditinal value.

Solve problem using sub query

Who is hired after Davies ?

To solve this problem you need two queries.

One query (Sub query)----When was Davies hired ?

Second query(Main query)---Find the name of employees who were hired after Davies ?

You can solve this problem by combining two queries, placing one query inside the other query.

The inner query(sub query) returns the value that is used by the outer query(main query)

The inner query(sub query) executes before the outer(main) query.

Syntax

select column1,column2,..... from table name where expression operator(select column1,column2,..... from tablename)

You can place sub query to WHERE, HAVING AND FROM clause.
Operator can be used like = , < , > etc.

Example

select last_name,hire_date from employees where hire_date>(select hire_date from employees where last_name='Davies');

Rules or Guidelines for using sub query

Enclose the sub queries in parentheses.
You can place subquery at the left or right side of operator but generally placed it at right side.
Use single row operator with single row sub queries and multi row operators in multi row sub queries.

Types of sub queries

Single row sub queries

It returns only one row from the sub query.

Multi row sub queries

It returns multiple rows from the sub query.

Multiple Column sub queries

It returns multiple column from the sub query.

Single row sub queries

Single row comparision operators

=, >, <, >=, <=, <>

Example

select last_name,job_id from employees where job_id=(select job_id from employees

where employee_id=141);

select last_name,job_id,salary from employees where job_id=(select job_id from employees where last_name='Davies') and salary>(select salary from employees where last_name='Davies');

Using group function in sub query

select last_name,job_id,salary from employees where salary=(select min(salary) from employees);

select last_name,job_id,salary from employees where salary=(select max(salary) from employees);

Having clause with sub query

select department_id,min(salary) from employees group by department_id having min(salary)>(select min(salary) from employees where department_id=30);

select job_id,avg(salary) from employees group by job_id having avg(salary)=(select min(avg(salary)) from employees group by job_id);

General errors in subquery

select employee_id,last_name from employees where salary=(select min(salary) from employees group by department_id); ---------//Error

To correct this error change = operator to in

select employee_id,last_name from employees where salary in(select min(salary) from employees group by department_id);

No rows return by the sub query.
Select last_name,job_id from employees where job_id=(select job_id from employees where last_name='Hass'); //no rows

Multi row sub queries

Milti row comaparision opeartors

IN Equal to any member in the list.

ANY Must be preceded by =, !=, >, <, >=, <= Returns TRUE if relation is true for at least one element exists in the result of sub query.

ALL Must be preceded by =, !=, >, <, >=, <= Returns TRUE if relation is true for all the elements exists in the result of sub query.

Example

select last_name,salary,department_id from employees where salary in(select min(salary) from employees group by department_id);

select employee_id,last_name,salary,job_id from employees where salary<any(select salary from employees where job_id='IT_PROG') and job_id<>'IT_PROG';

select employee_id,last_name,salary,job_id from employees where not salary<any(select salary from employees where job_id='IT_PROG') and job_id<>'IT_PROG';

<any means less than maximum
>any means more than minimum
=any is equivalent to in

select employee_id,last_name,salary,job_id from employees where salary<all(select salary from employees where job_id='IT_PROG') and job_id<>'IT_PROG';

select employee_id,last_name,salary,job_id from employees where not salary<all(select salary from employees where job_id='IT_PROG') and job_id<>'IT_PROG';

<all means less than minimum
>all means more than maximum
NOT opeartor can be used with IN ANY and ALL operators

Multiple columns subqueries

It return multiple columns to the outer query.
It can be pairwise or non parirwise comparision.

Syntax

select column1,columns2..... from table name where(column1,cloumns2....) in(select column1,column2..... from table name where condition)

Example

Pairwise Comparision

```
select first_name,salary,department_id from employees where (salary,department_id) in(select min(salary),department_id from employees group by department_id) order by department_id;

select employee_id,manager_id,department_id from employees where (manager_id,department_id) in(select manager_id,department_id from employees where employee_id in(174,199)) and employee_id not in(174,199);

select first_name,last_name,manager_id,department_id from employees where (manager_id,department_id) in(select manager_id,department_id from employees where first_name='Daniel');
```

Example

NonePairwise Comparision

```
select first_name,last_name,manager_id,department_id from employees where manager_id in(select manager_id from employees where first_name='Daniel') and department_id in(select department_id from employees where first_name='Daniel');

select employee_id,manager_id,department_id from employees where manager_id in(select manager_id from employees where employee_id in(174,199)) and department_id in(select department_id from employees where employeee_id in(174,199) ) and employee_id not in(174,199);
```

NULL values in sub query

Example

```
select e.last_name from employees e where e.employee_id not in(select m.manager_id from employees m);      ---No Rows because one of the value in subquery is NULL

select last_name from employees  where employee_id not in(select manager_id from employees  where manager_id is not null);      ---Right


select last_name from employees where employee_id in(select manager_id from employees);
```

Set Operators

Set operators combined the result of two or more queries into one result.
It is called compound queries.
All SET OPEARATORS have equal precedence.
SET OPEARATORS can be used in subqueries.

Set Operators Rules

No of columns(expressions) and datatype(same datatype group) in the second query must match the number of columns(expression) and data type of the first query.

ORDER BY clause can be placed at the end of whole statement not for individual query and it can recognizes only the columns of the first select query.

Duplicate rows are automatically eliminated except in **UNION ALL**.

Column names from first query appear in the result.

Output is sorted in assending order by default(First column of first select query) except in **UNION ALL**.

UNION ALL

Rows from both queries including duplications.

Example

```
select job_id from employees
union all
select job_id from job_history order by job_id;

select job_id,department_id from employees
union all
select job_id,department_id from job_history order by department_id;
```

UNION

Rows from both queries after eliminating duplications.

Example

```
select job_id from employees
union
select job_id from job_history;

select job_id,department_id from employees
union
select job_id,department_id from job_history;
```

INTERSECT

Rows that are common to both queries.

Example

```
select job_id from employees
intersect
select job_id from job_history;

select job_id,department_id from employees
intersect
select job_id,department_id from job_history;
```

MINUS

Rows in the first query that are not present in the second query.

Example

```
select job_id from employees
minus
select job_id from job_history;

select job_id,department_id from employees
minus
select job_id,department_id from job_history;
```

Matching Select Statement

You must match the data type when columns do not exists in one or other tables.

Example

select location_id,department_name "Department",TO_CHAR(null) "Warehouse Location" from departments union select location_id,TO_CHAR(null),state_province from locations;

```
select employee_id,job_id,salary from employees
union
select employee_id,job_id,0 from job_history;

select manager_id,department_id,hire_date from employees
union
select manager_id,department_id,to_date(null) from departments;
```

DML(Data Manululation Language) Statements(Insert, Update, Delete)

Insert

insert rows into the table.

Update

update existing rows in a table.

Delete

delete existing rows from the table.

Transaction

When you transfer money from account A to account B. It is called one transaction.

It contans three operations.

Update to account A (minus amount from A)

Update to account B (add amount to B)

Insert this details into third table C

Oracle sever must gurantee that all the three sql statements are performed to maintain the account in proper balance. When one operation from this transaction fails then other operations must be undone.

Insert

Insert rows into the table.

Syntax

insert into tablename(col1,col2.....) values(value1,value2....);

or

insert into tablename values(value1,value2....);

Example

insert into departments(department_id,department_name) values(1000,'hr');

In this syntax other two columns will automatically take null values.
You can keep any order of column but you have to place the values as the same order of column you specified.

insert into departemnts values(1000,'hr',null,null);
insert into departemnts values(1000,'hr','','');

In this syntax you have to explicitly palced null values in other columns.
You have to enter values as per the column order in the table.

Characters and date values must be entered in the single quotation marks.

Common errors while insterting

Missing values for mandatory **NOT NULL** columns.
Duplicate value violating any **UNIQUE** or **PRIMARY KEY** constraint.
Any value violating **CHECK** constraint
FOREIGN KEY constraint
Data type mismatch or values too wide to fit in a column

You can use in built function like **sysdate, current_date, to_date, lower, upper** etc to enter values in a table.

Example

Create table emp5(id number,name varchar2(20),hire_date date,salary number);

insert into emp5 values(1,initcap('jay'),'03-feb-03',15000);

insert into emp5 values(2,lower('KIRAN'),to_date('Feb 3, 2003','Mon dd,yyyy'),5000);

insert into emp5 values(3,upper('lalit'),sysdate,10000);

Creating script to insert multiple records

insert into emp5 values(&id,'&name','&hire_date',&salary);

or

insert into emp5(id,name,hire_date,salary) values(&id,'&name','&hire_date',&salary);

It will prompt to enter values every time.(**place / to run it)**

Copying rows from another table

You have to write insert statements with sub query.
Do not use values clause.
Match the no of columns in the insert clause to those in the sub query.
Insert all rows returned by the sub query

Example

create table sales_reps1(id number,name varchar2(30),salary number,commission_pct number);

or

create table sales_reps2(id,name,salary,commission_pct) as select employee_id, first_name, salary, commission_pct from employees where 1=2;

or

create table sales_reps3 as select employee_id id,first_name name,salary,commission_pct from employees where 1=2;

insert into sales_reps1(id,name,salary,commission_pct) select employee_id id,first_name

name,salary,commission_pct from employees where job_id like '%REP%';

insert into sales_reps2 select employee_id,first_name,salary,commission_pct from employees where job_id like '%REP%';

To create copy of rows In the table

This query will create an empty table with the same structure as employees table.

create table copy_emp12 as select * from employees where 1=2;

insert into copy_emp12 select * from employees;

To create copy of table with data

create table copy_emp13 as select * from employees;

Update

Update existing rows in a table.
It can update one row(**Primary key column in where condition**) or more than one row depends upon the condition(**other then primary key column**).

Syntax

update tablename set column=value, column=value..... where condition

Example

create table dept as select * from departments;
create table emp as select * from employees;

update emp set department_id=50 where employee_id=113;

update emp set department_id=50; (It will update all rows because no condition is given)

update emp set job_id='IT_PROG',commission_pct=null where employee_id=114;

Update two columns with multi column sub query

update emp set(job_id,salary) =(select job_id,salary from employees where employee_id=205) where employee_id=103;

Update two columns with two sub query

update emp set job_id =(select job_id from employees where employee_id=205),salary=(select salary from employees where employee_id=205) where employee_id=103;

Updating rows based on another table

update copy_emp set department_id=(select department_id from employees where employee_id=100) where job_id=(select job_id from employees where employee_id=200);

Delete

Delete existing rows from the table.

Syntax

delete from tablename where condition

Example

delete from dept where department_name='Finance';

delete from copy_emp; (all rows will be deleted because no where condition)

delete from emp where employee_id=114;

delete from dept where department_id in(20,40,50);

Deleting rows based on another table

delete from emp where department_id in(select department_id from dept where department_name like '%Public%');

TCL—Transaction Control Statements(commit,rollback,savepoint)

Syntax

commit

savepoint name

rollback

rollback to savepoint name

Example

Update
Update
Savepoint A
Delete
Delete
Savepoint B
Insert
Insert

rollback	All operations are undone
rollback to savepoint A	It will undo upto that savepoint
rollback to savepoint B	It will undo upto that savepoint
commit	Parmanently save data to database.

Points to be remember

Savepoint in not ANSI-standard SQL.

If you create second savepoint with the same name as the earlier savepoint, the earlier savepoint is deleted.

An automatic COMMIT occurs in the following circumtances

A DDL statement is issued.(create table,alter table,drop table,truncate table).
A DCL statement is issued.(grant,revoke).
Normal Exit from SQL DEVELOPER or SQL PLUS.
(selecting exit from file menu for SQL DEVELOPER or exit command for SQL PLUS).
Issuing COMMIT.

An automatic ROLLBACK occurs in the following circumtances

when there is an abormal termination of SQL DEVELOPER or SQL *PLUS or system

failure.(closing window by pressing cross button is abnormal exit).

AUTOCOMMIT command

Systex

set autocommit on;

set autocommit off;

State of data before COMMIT or ROLLBACK

The previous state of the data can be recovered.
The current session can view the result of the DML operations by using select statement.
Other sessions can not view the results of DML statements issued by the current session.
The affected rows are locked, other session can not change the data in affected rows.

State of data after COMMIT

Data changes are saved in the database.
The previous state of the data is overwritten.
All sessions can view the result.
Locks on the affected rows are released, those rows are available for other sessions to manipulate.
All savepoints are erased.

State of data after ROLLBACK

Data changes are undone.
Previous state of the data is restored.
Locks on the affected rows are released.

A DDL or DCL statement is automatically commited so it will end the pending temperory transaction.

Read consistency in oracle

Oracle gives the consistant view of data at all times.
Changes made by one user do not conflict with the changes made by another user.
Read consistency ensures that

Readers(select) do not wait for writers(insert,update,delete).
Writers do not wait for readers.
Writers wait for writers.

Database Objects

Table **It stores data in a row and column format.**
View
Sequence
Index
Synonym

Table naming rules

Table name and column name must

Begin with a letter.
Be 1 to 30 characters long.
0 to 9 , _ , $, # allowed.
It must be unique.
Reserved words are not allowed.
Table names and column names are not case sensitive.

DDL Statements(Data Definition Language)

Create table
Alter table
Drop table
Truncate table

create table syntax

create table tablename(col1 datatype,col2 datatype.....);

Datatypes in Oracle

varchar2(size)	Variable length character data.(minimum size is 1 maximum size you have to specified)
char(size)	Fixed length character data(minimum size is 1 maximum size 2000)
number(p,s)	Numeric data(p for precision(Total number of digits) and s for

	scale(number of digits after decimal point))
date	Date and time values
long	Variable length character data up to 2 GB
clob	Character large object up to 4 GB(Textbook)
blob	Binary large objects(images, audio,video file) up to 4 GB
bfile	Binary data stored in external file up to 4 GB
timestamp	Date and time values with fractional seconds
interval year to month	Interval of years and months
interval day to second	Interval of days, hours,minutes and seconds

Example

create table dept_copy(deptno number(2),dname varchar2(20),loc varchar2(20), create_date date default sysdate);

desc dept_copy; (You can view the structure of a table)

select table_name from USER_TABLES;

create table hire_dates(id number(8), create_date date default sysdate);

insert into hire_dates values(10,null);
insert into hire_dates(id) values(11);

Types of SQL constraints

NOT NULL	Null is not allowed.
PRIMARY KEY	Uniquely identifies each row of a table. Value must be unique. No duplicate value is allowed. No Null is allowed.
FOREIGN KEY	Values in the foreign key must match the values in the primary key table. (null is allowed,duplicate is also allowed)
UNIQUE	Value must be unique. No duplicate value is allowed. Null is allowed.
CHECK	Specifies the condition that must be true.

Column level constraint

Syntax

create table tablename(col1 datatype(size) constrainttype,col2 datatype(size) constrainttype,.....);

Example

create table emp123(id number(3) primary key,name varchar2(20) not null,email varchar2(30) unique,salary number(6) check(salary>5000),deptid number references departments(department_id));

If you do not provide the name for your constraint then oracle provides the constraint name that in the form of SYS_Cuniquenumber.

create table emp456(id number(3) constraint emp456pk primary key,name varchar2(20) constraint emp456notnull not null,email varchar2(30) constraint emp456uq unique,salary number(6) constraint emp456chk check(salary>5000),deptid number constraint emp456fk references departments(department_id));

Table level constraint

create table emp1234(id number(3),name varchar2(20) not null,email varchar2(30),salary number(6) ,deptid number,primary key(id),unique(email),check(salary>5000),foreign key(deptid) references departments(department_id));

If you do not provide the name for your constraint then oracle provides the constraint name that in the form of SYS_Cuniquenumber.

create table emp4567(id number(3),name varchar2(20) not null,email varchar2(30),salary number(6) ,deptid number,constraint empemp4567pk primary key(id),constraint empemp4567uq unique(email),constraint emp4567chk check(salary>5000),constraint empemp4567fk foreign key(deptid) references departments(department_id));

On delete cascade and On delete set null in Foreign key

On delete cascade

Deletes the dependent rows in the child table when a row in the parent table is deleted.

On delete set null

Set dependent foreign key values to null when a row in the parent table is deleted.

Points to be remember

Not null constraint can be defined at the column level only.
You can define more than one constraint on a particular column.
Composite constraints(on combination of more than one column) can be defined on table level only(composit primary key, composit unique key, composit foreign key)
You can define only one primary key in a table.
You can define more than one foreign key in a table.
The primary key and foreign key can be in the same table or can be in a different table.

Create table using subquery

Syntax

create table tablename(col1,col2.....) as subquery;

match the number of specified columns to the number of subquery columns.
If no columns are specified the columns names of the table are same as the column names in the subquery.
The column datatype definition and not null constraint are passed to the new table.

Example

create table dept80 as select employee_id,last_name,salary*12 annsal,hire_date from employees where department_id=80;

The expression salary*12 must be given an alias like annsal otherwise it will give error.

create table dept80(id,name,annsal,hire_date) as select employee_id,last_name,salary*12, hire_date from employees where department_id=80;

Truncate

It removes all rows from the table.
It makes the table empty.
It can not be easily undone because it is DDL statement.
It is more effficient then delete statement because truncate statement is a DDL statement and generates no rollback information.
It quickly remove all rows from a table.
If the table is a parent of referential integrity constraint(foreignk key), you can not truncate table.

You can disable the constraint and truncate it.

Syntax

truncate table tablename;

Example

truncate table copy_emp;

Alter table

Use alter table statement to modify the structure of a table like

Add new column
Modify an existing column definition
Define a default value for a new column
Drop a column
Rename a column
Change table to read only

Syntax

alter table tablename add(column1 datatype default value,column2 datatype.....);

alter table tablename modify(column1 datatype default value,column2 datatype.....);

alter table tablename drop(col1,col2.....); **(To drop more than one columns)**

alter table tablename drop column columnname; **(To drop only one column)**

Example

alter table dept80 add(job_id varchar2(20));

alter table dept80 modify(last_name varchar2(30));

alter table dept80 drop(job_id);

Points to be remember

The new column becomes the last column.

If a table already contains rows when a column is added, the new column is initially null or takes the default values for all rows.

You can add a mandotory not null column to a table that contains data in other columns only if you specify default value.

You can add a not null column to an empty table without default value.

A change to the default value affects only subsequent insertions to the table.

You can increase the width and precision of a numeric column.

You can increase the width of a character column.

You can decrease width of column if

Column contains only nulll values

The table has no rows

Decrease in column width is not less than the existing values in that column.

You can change the datatye if column contains only null values.

CHAR TO VARCHAR2 conversion can be done with a data in a column

You can convert CHAR TO VARCHAR2 OR VARCHAR2 TO CHAR only if the column contains null values or if you do not change the size.

Set table Read Only

You can make table read only for maintanance.

You can not perform DML operations during maintanance when table is read only.

You can drop the read only table.

Example

alter table employees read only;

alter table employees read write;

Manage Constraints

Adding and droping Constraints

Enabling and disabling the constraints

You can add a **not null** constraint by **modify** clause.

You can add not null constraint if table is empty or if a column has a value for every row.

Syntax

alter table tablename add constraint constname type(columnname);

alter table tablename drop constraint constname;

Example

alter table emp2 modify employee_id primary key;

alter table emp2 add constraint emp2pk primary key(employee_id);

alter table emp2 add constraint emp2fk foreign key(manager_id) references emp2(employee_id);

alter table emp2 drop constraint emp2fk;

alter table emp2 drop primary key cascade;
(It will remove primary key as well as associated foreign key)

alter table emp2 drop column employee_id cascade constraints;
(It will remove employee_id column,primary key as well as associated foreign key)

Rename table, column and constraints

Example

alter table emp2 rename to newemp2;

alter table emp2 rename column commission_pct to comm;

alter table emp2 rename constraint emp2fk to newemp2fk;

Enable and disable constraints

alter table emp2 disable constraint emp2pk;

alter table dept2 disable primary key cascade;
(It will disable primary key as well as associated all foreign keys)

alter table emp2 enable constraint emp2pk;

Drop table

To remove table from database.
It will move the table in to recycle bin so table space will not be released.
If **purge clause** is used then table is removed parmanently so table space will be released.

Example

drop table dept80;

select * from recyclebin;

purge recyclebin;

drop table emp purge;

flashback table emp to before drop;

Database Object Sequence

To generate numeric values automatically.
Many applications require use of unique numbers as primary key values. You can either build code in to the application or use a sequence to generate unique numbers.
It is a sharable object among multiple users.
Sequence values can be cached in to the memory so fast accessible.
One secuence object can be used for multiple tables.

Syntax

create sequence squencename
start with or increment by number
maxvalue number or nomaxvalue
minvalue number or nominvalue
cycle or nocycle
cache number or nocache

Points to be remember

If you omitt start with then it will start from 1.
If you omitt increment by then it will increment by 1.
NOCYCLE is default if you do not specify.
CYCLE means after reaching maximum or minimum it will start again.

Cache means how many values you want to generate and keeps in cache.
By default oracle generates 20 values in cache.

Example

create sequence dept_deptid_seq start with 10 increment by 10 maxvalue 999 nocache nocycle;

NEXTVAL AND CURRVAL

Nextval returns the next available sequence value.
Currval obtains the current sequence value.
Nextval must be issued before the currval.

Syntax

sequencename.nextval
sequencename.currval

You can use sequence values in select ,insert and update statement.
You can also use to create default values in create table statement.

Example

select dept_deptid_seq.nextval from dual;

select dept_deptid_seq.nextval,dept_deptid_seq.currval from dual;

insert into dept values(dept_deptid_seq.nextval,'hr',100,1700);

create sequence s1 start with 1;

create table emp(id number default s1.nextval not null,name varchar2(20));

insert into emp(name) values('jay');

insert into emp(name) values('kiran');

select * from emp;

Gap in sequence values

A gap in sequence values occur when

A rollback occurs.
A system crashes.
It is used in more than one table.

How to Modify the sequence

alter sequence dept_depid_seq increment by 20 maxvalue 9999 nocache nocycle;

Points to be remember

Only future sequence numbers are affected.
The sequence must be dropped and recreated to restart a sequence with different number.
Some validation is performed like maxvalue can not be less than current sequence number.

How to remove sequence

drop sequence dept_deptid_seq;

DataBase Object Synonym

It is an alternative name to any object.
It is usefull in hiding the identity and location of an object.
It will give shorter name to lengthy object name.

Sytntax

create synonym synonymname for object;

create public synonym synonymname for object;
(create synonym that is accessible to all users. Only DBA can create public synonym)

Example

create synonym d_sum for dept_sum_vu;

drop synonym d_sum;

create public synonym dept for hr.departments;

drop public synonym dept;

Database Object Index

It can be used by orale server to speed up the retrival of rows.
It is used and maintaned automatically by oracle server.
It can be created implicitly(automatically) or explicitly by user.
If you do not have index on column than full table scan occurs.
You have to just create the index nothing to do with it because oracle server uses this index.
When you drop the table corresponding indexes are also droped.

How to create index object

Implicitly or Automatically

A unique index is created automatically when you define a PRIMARY KEY OR UNIQUE constraint in a table.

Manually or Explicitly

You can create unique or nonunique index on columns to speed up acess to the rows.

Two types of index

Unique Index

A unique index is created automatically when you define a PRIMARY KEY OR UNIQUE constraint in a table. The name of this index is same as the constraint name.
You can also create unique index manually.

Nonunique Index

A user can create nonunique index like an index on foreign key column to speed up the retrival of rows.

Syntax

create unique or nonunique(Default) index indexname on table(col1,clo2.....);

Example

create index emp_lastname_idx on employees(last_name);

Create the index with create table statement

create table new_emp(emp_id number(6) primary key using index(create unique index emp_id_idx on new_emp(emp_id)),first_name varchar2(20));

create table new_emp2(emp_id number(6) primary key,first_name varchar2(20));

Example

create table new_emp3(emp_id number(6) ,first_name varchar2(20));

create unique index emp3_id_idx on new_emp3(emp_id));

alter table new_emp3 add primary key(emp_id) using index emp3_id_idx;

Removing an Index

Syntax

drop index indexname;

drop index emp3_id_idx;

Database Object View

It actually represents subset of data from one or more tables.
By using viwes you can present or hide data from the table.
It is a window through which data from tables can be viewed or changed.
It is just a select statement it does not have it's own data.
The tables on which a view is based is called base table.
It is used

To restrict data access
To make complex queries easy
To present different views of the same data.

Types of views

Simple View

Number of tables	One
Contain funtions	NO
Contain groups of data	NO
DML Operation through a view	Yes

Complex View

Number of tables	One or More
Contain funtions	Yes
Contain groups of data	Yes
DML Operation through a view	Not Always possible

Syntax

(create or replace) (force/noforce) view viewname(alias1,alias2.....) as subquery

replace	Re-create the view if already exists.
force	Create a view regardless of base table exists or not
noforce	Create a view only if base table exists.(default)
alias	Specifies the names for the expression selected by the subquery (no of alias must match the no of expression in subquery)

Example

create or replace view empvu80 as select employee_id,last_name,salary from employees where department_id=80;

desc empvu80;

select * from empvu80;

create or replace view salvu55 as select department_id dept_id,employee_id id_number,last_name name,salary*12 annsal from employees where department_id=55;

desc savu50;

select * from salvu50;

create or replace view salvu55(department_id,id_number,name,annsal) as select department_id,employee_id ,last_name,salary*12 from employees where department_id=55;

desc savu55;

select * from savu55;

How to modify view

create or replace view empvu80(id_number,name,sal,dept_id) as select employee_id, first_name||' '||last_name,salary,department_id from employees where department_id=80;

Complex veiw

create or replace view dept_sum_vu(name,minsal,maxsal,avgsal) as select department_name,min(salary),max(salary),avg(salary) from employees join departments using(department_id) group by department_name;

desc dept_sum_vu;

select * from dept_sum_vu;

DML operations on view

You can perfom DML operations on simple view.

You can not remove rows if view contains the following
- group functions
- gorup by clause
- distinct keyword

You can not modify or update data if view contains the following
- group functions
- gorup by clause
- distinct keyword
- Expressions like salary*12

You can not insert data if view contains the following
- group functions
- group by clause
- distinct keyword
- Expressions like salary*12
- not null column without default value in a base table that are not selected by the view.

Removing view

Syntax

drop view viewname;

drop view empvu10;

How to create users

You can control the database access to specific objects and add new users with different level of access permissions.

Database security

It can be classified in two categories.

System security

It covers access and use of database as the system level, such as the username and password, the disk space allocated to users, and the system operations that user can perform.

Data security

It covers access and use of database objects and the action that those users can perform on the objects.

Points to be remember

A privilege is the right to execute the particular sql statements.

DBA is a high level user, which creates the user and grant users access to the databae and it's objects.

Users requires the system privileges to gain access to the database and object privileges to manipulate the content of the objects in the database.

Users can also be given the privilege to grant additional privilegs to other users or role.

Schema is collection of objects such as tables,views and sequences.

The schema owned by the database user and has the same name as that user.

System privileges

More than 200 privileges are available.
It is provided by the DBA.

DBA Has the following privileges

Create user
Drop user
Drop any table
Back up any table
Select any table
Create any table

Syntax

create user username identified by password;

Example

create user demo identified by demo;

Grant system privileges

After the user is created, the DBA can grant specific system privileges to that user.

Grant privi1,privi2..... to user1,user2.... or role or public

An application developer may have these system privileges

Create session
Create table
Create view
Create sequence
Create procedure
Create function

Example

grant create session,create table,create view,create sequence to demo;

Role

A role is named group of related privileges that can be granted to the user.
This method makes it easier to revoke and maintain privileges.

A user can have access to several roles and several users can be assigned to the same role.
First DBA must create a role, the DBA can assign privileges to the role and assign the role to the users.

Syntax

create role rolename;

Example

create role manager;

grant create session,create table,create view to manager;

grant manager to alice;

How to change your password

alter user demo identified by demo2;

or

passw or password command in SQL PLUS

Object privilegs

Alter table,sequence
Delete table,view
Index on table
Insert in table or view
Select table or view or sequence
Update on table or view

It varies from object to object.
An owner has all the privileges on the object.
An owner can give specific privileges on that owner's object.

Syntax

grant all or object privileges(column) on object to (user/role/public) with grant option;

all specifies all object privileges

public	To all users
with grant option	Enables the grantee to grant to other users and roles.

Object privileges granted with the with grant option clause are revoked when the grantor's privilege is revoked.

Example

grant select on employees to demo;

grant update(department_name,location_id) on departments to demo,manager;

grant select,insert on departments to demo with grant option;

grant select on departments to public;

Revoking the privileges

Sytanx

revoke (privi1,privi2..... or all) on object from (user1,user2... or role or public)

Example

revoke select,insert on departments from demo;

Data Dictionary views

It is used to retrieve metadata about your schema objects.

User tables contain business data such as employees,departments etc.

There is another collection of tables and views in oracle database known as data dictionary.

This collection is created and maintain by oracle server and contains information about the database.

It is very important for all users like application developer,application desginers,database administrator

It is read only so you can issue only select statement.

The names in the data dictionary are in uppercase.

You can find following inforamtion

Definitions of all schema objects(Tables,Views,Indexes,Synonyms etc).
Default values for column
Constraint information
Names of oracle users
Privileges and roles that each user has been granted.
Other general database information.

Views Prefix

user User's view(What is in your schema, what you own)

Example

USER_OBJECTS

USER_OBJECTS

DESC USER_OBJECTS;

SELECT OBJECT_NAME,OBJECT_TYPE,CREATED, STATUS FROM USER_OBJECTS ORDER BY OBJECT_TYPE;

SELECT * FROM CAT;

SELECT * FROM USER_CATALOG;

Get the table information(USER_TABLES)

DESC USER_TABLES;

SELECT TABLE_NAME FROM USER_TABLES;

SELECT TABLE_NAME FROM TABS;

Get the column information(USER_TAB_COLUMNS)

DESC USER_TAB_COLUMNS

SELECT COLUMN_NAME, DATA_TYPE, DATA_LENGTH, DATA_PRECISION, DATA_SCALE, NULLABLE FROM USER_TAB_COLUMNS WHERE TABLE_NAME='EMPLOYEES';

Get constraint information(USER_CONSTRAINTS)

DESC USER_CONSTRAINTS

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION, R_CONSTRAINT_NAME, DELETE_RULE, STATUS FROM USER_CONSTRAINTS WHERE TABLE_NAME='EMPLOYEES';

C CHECK OR NOT NULL
P PRIMARY KEY
U UNIQUE KEY
R FOREIGN KEY

Get constraint column information(USER_CONS_COLUMNS)

DESC USER_CONS_COLUMNS;

SELECT CONSTRAINT_NAME, COLUMN_NAME FROM USER_CONS_COLUMNS WHERE TABLE_NAME='EMPLOYEES';

Get sequence information(USER_SEQUENCES)

DESC USER_SEQUENCES

SELECT SEQUENCE_NAME, MIN_VALUE, MAX_VALUE, INCREMENT_BY, LAST_NUMBER FROM USER_SEQUENCES ;

Get synonym information(USER_SYNONYMS)

DESC USER_SYNONYMS;

```
SELECT * FROM USER_SYNONYMS;
```

Get index information(USER_INDEXES)

```
DESC USER_INDEXES;

SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS FROM USER_INDEXES
WHERE TABLE_NAME='EMPLOYEES';
```

Get index column information(USER_IND_COLUMNS)

```
DESC USER_IND_COLUMNS;

SELECT INDEX_NAME,COLUMN_NAME, TABLE_NAME FROM
USER_IND_COLUMNS WHERE INDEX_NAME='LNAME_IDX';
```

Get views information(USER_VIEWS)

```
DESC USER_VIEWS;

SELECT VIEW_NAME FROM USER_VIEWS;

SELECT TEXT FROM USER_VIEWS WHERE VIEW_NAME =
'EMP_DETAILS_VIEW';

SET LONG 1000
```

Interval Data type

1)Interval year(Year precision) to month(Year, Month)

Year precision is the no of digits in the year datetime field.
Accepted values from 0 to 9.
The default is 2.

2)Interval day(Day precision) to second(Fractional second precision)(Day, Hour, Minute, Second)

Day precision is the no of digits in the day datetime field.
Accepted values from 0 to 9.
The default is 2.

Fractional precision is the no of digits in the fractional part of the second datetime field.

Accepted values from 0 to 9.
The default is 6.

Year	**Any positive or negative number**
Month	**00 to 11**
Day	**Any positive or negative number**
Hour	**00 to 23**
Minute	**00 to 59**
Second	**00 to 59**

Example

```
create table warranty(prod_id  number,warranty_time interval year(3) to month);

insert into warranty values(10,interval '8' month);

insert into warranty values(11,interval '200'  year(3));

insert into warranty values(12,'200-11');

insert into warranty values(12,interval '123-2' year(3) to month);

insert into warranty values(12,interval '123' year(3));

insert into warranty values(12,interval '300' month(3));

select * from warranty;
```

Example

```
create table lab(exp_id number,test_time interval day(2) to second);

insert into lab values(1,'90 00:00:00');

insert into lab values(2,interval '5 03:10:30' day to second);
```

Datetime functions

Syntax

Extract(Year or Month or Day or Hour or Minute or Second from Expression)

To extract year, month, day, hour, minute etc. component from datatime values.

Example

```
select last_name,employee_id,hire_date from employees where extract(year from
to_date(hire_date,'DD-MON-RR'))>2007 order by hire_date;

select last_name,hire_date,extract(month from hire_date) from employees where
manager_id=100;
```

Syntax

Tz_offset(Timezone name or offset(+|- hh:mi) or Sessiontimezone)

It returns the timezone offset for correstponding to the value entered.

Example

```
select tz_offset('Us/Eastern'),tz_offset('Canada/Yukon'),tz_offset('Europe/London') from
dual;

select * from V$TIMEZONE_NAMES;
```

Syntax

To_yminterval()

To convert char string value to interval year to month datatype.

Example

```
select hire_date,hire_date+to_yminterval('01-02') as hiredateyminterval from employees
where department_id=20;
```

Syntax

To_dsinterval()

To convert char string value to interval day to second datatype.

Example

```
select last_name,to_char(hire_date,'mm-dd-yy hh:mi:ss'), to_char(hire_date
+to_dsinterval('100 10:00:00'),'mm-dd-yy hh:mi:ss') from employees;
```