INDEX

1. INTRODUCTION

1.1 Database
A database is an organized collection of related information.

1.2. Relational Database
A Relational database is a collection of relations or two-dimensional tables.

1.2.1. Entity:
A thing of significance about which information needs to be knows. Examples are Departments, Employees, and Orders.

1.2.2 Attribute:
Something that describes or qualifies an Entity. For examples, for the employee entity the attribute would be the employee number, name, job title, Dept number and so on.

Relationship:
A named Association between entities.

1.3 Relating Multiple Tables
Each table contains data that describes exactly one entity. Because Data about different entities is stored in different tables, you may need to combine two or more tables to answer a particular question. For example, you may want to know the location of the dept where an employee works. In this scenario you may need information from employees table, which contains data about employees, and the Department table, which contains data about department with an RDBMS you can relate the data in one table to the data in another table by using the Foreign Keys a Foreign Key is a column or a set of columns that refer to a primary key in the same table or another table.

Relations among data:
A Relationship is defined, as “An association among entities”. A relationship type is an association of entity types, while a relationship instance is an association of entity instance. A relationship may associate an entity with itself. Several relationships may exist between the same entities. The three different types of relationships recognized among various data stored in the database are:

- One-to-one
- One-to-Many (or many-to-one)
- Many-to-Many

1.4 Database Models:

The purpose of data models is to communicate, categorize, describe, specify, investigate and analyze data. Database Models are broadly classified into two categories. They are:

- Object-Based logical models
- Record-based logical models

The Object-based logical models can be defined as a collection of ideal tools for describing data, data relationships and data constraints. The object-based logical models can be defined as a collection of ideal tools for describing data, data relationships. Application. A structure once created can be reused-this is the fundamentals property of the OOPs concepts. By combing the object oriented and relational concepts Oracle now offers the best of both worlds.

1.5 Codds Rules

DR. E. F. Codd first outlined the Principles of the relations model in 1970 paper called "A Relational Model of Data for large Shared Data Banks." In this paper, Dr. Codd proposed the relational model for database systems. Certain good rules have to be followed for a DBMS to be relational theory to its full potential. These twelve rules are listed below.

- **The information Rule**
  All information is explicitly and logically represented in tables as data values.

- **The Rule of Guaranteed Access**
  Every item of data must logically addressable with the help of a table name, Primary key value and column name.

- **The Systematic Treatment of null values**
  The RDBMS must be able to support null values to represent missing or inapplicable information.

- **The Database Description Rule**
  A Description of database is maintained using the same logical structures with which data was specified by the RDBMS. These are accessible to users with appropriate authority and are stored in the data dictionary.

- **Comprehensive Data Sub Language**
  According to this rule, the system must support Data definition, view definition, Data manipulation, integrity constraints, Authorization, Transaction management operations.

- **The View Updating Rule**
  All the views must be updatable by the system

- **The Insert and Update Rule**
  This Rule specifies that all the data manipulation commands must be operational on sets of rows having a relation rather than on a single row.

- **The Physical Independence Rule**
  Application programs must remain unimpaired when any changes are made in storage representation or access methods.

- **The Logical Data independence Rule**
  The changes that are made should not affect the users ability to work with the data. The change can be splitting the table into many more tables.

- **The Integrity Independence Rule**
  The integrity constraints should be stored in the system catalog or in the Database as tables.

- **The Distribution Rule**
  The system must be able to access or manipulate the data that is distributed in other systems.

- **The Non_Subversion Rule:**
  The non_subversion rule states that different levels of the language cannot subvert or Bypass the integrity rules and constraints. If an RDBMS support a lower level language then it should not bypass any integrity constraints defined in the higher level.

1.6 The Relational Model

The relational model has three major characteristics:

Structures	Structures are well-defined objects (such as tables views, indexes, and so on) that store or access the data of a database. Structures and the data contained within them can be manipulated by operation
Operations	Operations are clearly defined actions that enable users to manipulate the data and structures of a database. The operations on a database must add here to a predefined set of integrity rules.
Integrity rules	Integrity rules are the laws that govern which operations are allowed on the data and structures of database integrity rules protect the data and the structures of a database.

Relational Database Management System

Oracle provides a flexible RDBMS called oracle9i. Using its features, you can store and manage data with all the advantages of relational structure. plus PL/SQL, an engine that provides you with the ability to store and execute program units.

The Oracle server offers the option of retrieving data based on optimization techniques. It includes Security features that control how a database is accessed and used. Other features include consistency and protection of data through locking mechanism.

1.7 What is Oracle?

Oracle consists of comprehensive set of application building and end-user products, aimed at providing complete information technology solution. Oracle's applications are portable across a wide range of platforms and operating system from PC's to large parallel processors. Oracle Application may run on the same computer as the Oracle Family there is a common ability to access the database. Whether directly or indirectly, this is achieved through the structured query language.

1.8 The Object-Relational Model

The Object-relational model enables users to define object types, specifying both the structure of the data and the methods of operating on the data, and to use these datatypes within the relational model.

An object type differs from native SQL datatypes in that it is user-defined and it specifies both the underlying persistent data (attributes) and the related behaviors (methods).

Object types are abstraction of the real-world entities—for example, purchase orders -- that application programs deal with. Object types are used to extend the modeling capabilities provided by the native datatypes.

An object type has three kinds of components:

- A name, which serves to identify the object type uniquely.
- Attributes, which are build-in datatypes or other user-defined types. Attributes model the structure of the real world entity.
- Methods, which are functions or procedures written in PL/SQL and stored in the database, or written in a language such as C or Java and stored externally. Methods implement specific operations that an application can perform on the data. Every object type has a constructor method that makes a new object according to the datatype's specification.

1.9 The Object- Relational Model for Database Management

Database management systems have evolved from hierarchical to network to relational models. The most widely accepted database model is the relational model. Oracle extends the relational model to an object relational model which makes it possible to store complex business models in a relational database.

Relational database management systems offer benefits such as:
- Intendance of physical data storage and logical database structure
- Variable and easy access to all data
- Complete flexibility in database design
- Reduced data storage and redundancy

1.10 Oracle 9i Database New Features

Oracle 9i offers a comprehensive high-performance infrastructure for e-business. Oracle 9i includes everything needed to develop, deploy, and manage internet applications. Oracle9i include Scalability from departments to enterprise e-business sites. One development model, easy development options and one management interface for all application. Oracle9i are two products, Oracle 9i Application Server and Oracle9i Database that provide complete and single infrastructure for internet applications. Oracle 9i runs all your application and 9i database stores all your data. This is not just the object relational data you expect an enterprise database to manage. It can also be unstructured data like spreadsheets, word documents and PowerPoint Presentation, xml and Multimedia data types like mp3, graphics, videos and more. Oracle9i takes business where it needs to be meeting and exceeding stringent demands for high-quality service in a service-driven marketplace. Oracle9i is design to optimize traditional, internet and intranet applications, and to stimulate the emerging hosted application market on the internet. Oracle9i builds on historic strengths to offer the first complete and simple software infrastructure for the internet's next generation of intelligent, collaborative applications. The Oracle9i new features expedite delivery of critical performance, scalability, and availability essential to providing hosted service software for anyone, anywhere, facilitate critical advancement for internet- based business, and stimulate the emerging hosted application market. New Oracle9i Database features deliver the performance, scalability, and availability essential to hosed service software made available to anyone, anywhere. The Oracle9i Database offers new transparent, rapid growth clustering capabilities, along with powerful and cost-effective security measures, zero- data- loss safeguards, and real- time intelligence to deliver the power needed in today's dynamic marketplace.

Important new features are introduced with the release of the Oracle9i Database, including advancements in Real Application Clusters, system management, availability, scalability, and security.

Availability
Oracale9i dramatically extends Oracle's leadership in Internet database availability, which is critical for any e-business application. Key areas in Oracle9i include:

Enhanced Disaster Recovery Environment
Many tasks for managing a standby database are now automated. Log Miner now supports log queries based in changes and has comprehensive log analysis for additional data types, plus an easy- to- use graphical user interface.

Online Data Evolution
Oracle9i contains a new online reorganization and redefinition architecture that allows much more powerful reorganization capabilities, including an online CREATE TABLE AS SELECT operation.

Precision Database Repair
Oracle9i includes better prevention and improves handling of disk corruption, and is table the database to a consistent state after recovery.

Scalability
Oracle9i allows e-business to scale the most demanding e-business applications with intensive transactions loads. Key areas include: Scalability opportunities are increased with little or no performance cost through Oracle9i Real Application Clusters. Customers can horizontally scale the database tier as needed.

Fine Graine, Automatic Resource Management
Administrator's gain more granular control over resources through new features, and can specify the maximum active sessions per consumer group.

Native Compilation and Improves PL/SQL Optimization

Support for native compilation and optimization of PL/SQL improves performance of many business applications.

Security
Oracle9i continues to provide the most secure application development and deployment platform in the industry. Three-tire security is enhanced by proxy authentication, including credential proxy of x.509 certificates or distinguished names (DN), support for thick JDBC, Connection pooling for application users (thick and thin JDBC and OCI), and integration with LDAP. Hosting security is provided through enhancements in Virtual Private Database (VPD), fine- grained auditing, and Web-based single sign-on. A large number enhancement has been made in areas ranging from management of network naming and password based user management to new replication queue management and reconciliation tools.

1.11 Database Users and Schemes

Each oracle database has a list of usernames. To access a database, a user must use a database application and attempt a connection with valid username of the database. Each username has an associated password to prevent unauthorized use.

Schema Objects
A schema is a collection of logical structures of data, or schema objects. A schema is owned by a database user and has the same name as that user. Each owns a single schema. Schema objects can be created and manipulated with SQL and include the following types of objects:

- Clusters
- Constraints
- Database links
- Database triggers
- Dimensions
- External procedure libraries
- Index-organized tables

- Index
- Index types
- Java classes, Java resources, java sources
- Materialized views
- Materialized view logs
- Object tables
- Object types
- Object views
- Operators
- Packages
- Sequence
- Stored functions, stored procedures
- Synonyms
- Tables
- Views

Non Schema Objects

Other types of object are also stored in the database and can be created and manipulated with SQL but are not contained in a schema:

- Contexts
- Directories
- Parameter files (PFILEs) and server parameter files (SPFILEs)
- Profiles
- Roles
- Rollback segments
- Table spaces
- Users

1.12 Oracle Product Includes:

SQL, i SQL *plus and PL/SQL
SQL: structured Query Language is used to access a relational database including Oracle. IT may be used with each of the Oracle tools where access to the database is required.
The oracle server supports ANSI standard SQL. Statements manipulate data and data definitions in the database.

SQL *plus: It is an Oracle Product with which SQL and PL/SQL language may be used. It has its own command language for controlling the behavior of the product, and for formatting the output from SQL Queries server for execution and contain its own command language ISQL * plus is an environment which is oracle proprietary. It runs on the browser and access local and remote database. These command do not allow manipulation of values in the database.

PL/SQL: It is the Oracle's Procedural Language for writing application logic and for manipulating data out side the database. It includes sub-set of SQL commands and is available in the Oracle Server itself.

1.13 SQL Supports the Following Categories of Commands:

Data Definition Language	- Create, alter, drop, truncate, rename commands
Data Manipulate Language	- Insert, select, delete and update commands
Data Control Language	- grant and revoke commands
Transaction Control Language	- Commit, savepoint and rollback commands.

A SQL statement can be thought of as a very simple, but powerful, computer program or instruction. SQL statements are divided into the following categories:

Data Definition Language (DDL) Statements
Data definition language statements define, maintain, and drop schema objects when they are no longer needed. DDL statements also include statements that permit a user to grant other users the privileges, or rights, to access the database and specific objects within the database.

Data Manipulation Language (DML) statements
Data manipulation language statements manipulate the database's data. For example, querying, inserting, updating and deleting rows of a table are all DML operations. Locking a table or view and examining the execution plan of an SQL statement are also DML operations.

Data Control language (DCL) statements
Data control language provides users with privilege commands. The owner of database objects, say Tables, has the sole authority over them. The owner can allow other database users access to the object as per his/her discretion. Granting privilages (insert, select) to other allows them to perform operation within their (privilages) Purview. Privilages granted can also be withdrawn by the owner any time.

Transaction Control Statements
Transaction control statement manage the changes made by DML statements. They enable the user or application developer to group changes into logical transactions. Examples include COMMIT, ROLLBACK, and SAVEPOINT.

1.14 Transactions

A transaction is a logical unit of work that comprises one or more SQL statements executed by a single user. According to the ANSI/ISO SQL standard, with which oracle is compatible, a transaction begins with the user's first executable SQL statement. A transaction ends when it is explicitly committed or rolled back by that user.

Consider a banking database. When a bank customer transfers money from savings accounts to a checking account, the transaction can consist of three separate operations: decrease the savings account, increase the checking account, and record the transaction in the transaction journal.

Oracle must guarantee that all three SQL statements are performing to maintain the accounts in proper balance. When something prevents one of the statement in the transaction from executing (such as a hardware failure), the other statements of the transaction must be undone; this is called rolling back. If an error occurs in making any of the updates, then no updates are made.

1.15 Basic SELECT Statement

Select * |{ [distinct] column | expression [alias],..}from table;
A select clause specifies the columns to be displayed. A from clause specifies the table name.

- Asterisk (*) Selects all columns from the table
- Distinct suppress duplicates

Selecting all columns

```
SQL> select * form emp;
SQL> select * from dept;
```

Selecting specific columns

```
SQL> select empno, ename from emp;
SQL> select deptno, dname from dept;
```

Column Label Defaults
Label default justification:
- Left: date and character data
- Right: numeric data

Label default display is uppercase

Arithmetic Expression
You can change the way the data is displayed, perform calculation or look what if scenarios. We can create expressions with number and date data by using arithmetic operators '+' Add, '–'subtract, '*' multiply and '/' divide.

```
SQL> select ename, sal, sal*200 from emp;
SQL> select ename, sal, 10*sal+300 from emp;
SQL> select ename, sal, 10*(sal+300) from emp;
```

If an arithmetic expression contains more than one operator, multiplication and division are evaluated first. If the operator within an expression are of same priority, then evaluation is done from left to right you can use parentheses to force the expression within the parentheses to be evaluated first.

Defining a Null Value:
A null is a value that is unavailable, unassigned, unknown or inapplicable. A null is not same as zero or a blank space.

```
SQL>Select ename, job, sal, comm from emp;
```

Note: Arithmetic Expression containing a null value evaluated to null.

For example if you attempt to perform division with zero. You will get an error However, if you divide by null, the result is null.

SQL> select ename, 10*(sal+comm.) from emp;

Managing Null Values
If a row lacks a data value for a particular column, that value is said to be null, or to contain null. A null value is not the same as Zero or a space. Zero is a number, and a space is a character. Null values take up one byte of internal "Storage" overhead.

Defining Column Alias
When displaying the result of a query, SQL*plus normally uses the selected column's name as the heading. In many cases, that heading may be difficult to understand or even meaningless. You can change a column's heading by using a column alias. A column alias renames the column heading. It requires double quotation marks. If it contains spaces or special characters or is case sensitive.

SQL> select ename "employee name", job, sal salary from emp;

Concatenation operator
A concatenation operator concatenation columns or character strings to other columns. Two vertical bars represent it (||).

SQL>select ename || job as "Employees info" from emp;

Literal Strings:
A Literal String is any user defined string in the 'Select Statement'.

```
SQL> Select ename || ' works as' || 'job' || ' with a salary of :' || sal "Emp Details"
         From emp;
```

Eliminating Duplicate Rows
Eliminate duplicate rows by using the distinct keyword in the Select Clause.

SQL> select distinct deptno from emp;

DISTINCT with Multiple Columns

SQL> Select distinct deptno, job from emp;

You can specify multiple column after the DISTINCT qualifiers. The DISTINCT qualifiers affect all selected columns. When DISTINCT is applied to multiple column, The result represents the distinct combination of the columns.'

Displaying Table Structure

SQL>DESCRIBE emp;

Questions

1. What are the features of Isql*plus.
2. List the different job available in the emp table.
3. Where are the different categories of SQL commands.
4. Display all columns and all rows of dept table.
5. Create a query the ename, job, hiredate and empno for all employees. Provide an alias STARDATE for the hiredate column.

2. RESTRICTING AND SORTING DATA

Restrict the rows returned by using the 'Where' clause. The where clause follows the from clause. We can retrieve only the rows which satisfy the 'Where' condition. To arrange the display rows. According to some pre-defined order we can use the 'order by' clause. It is also use to arrange rows in the ascending and descending order.

Character strings and dates in the WHERE clause must be enclosed in a single Quotation marks (''), Number constants, however, must not.

E.g. Display all the employees of department 30.
SQL>select *from emp where deptno=30;

E.g. Display all details of Scott.
SQL> Select * from emp where ename='SCOTT';

2.1. Comparison operators

Comparison conditions are used in conditions that compare one expression to another value or expression logical comparison operators.
= equal to,
> greater than,
< less than,
>= greater than equal to
<= less than equal to,
<> not equal to

SQL comparison operators

Between--- and ---

Between two values
E.g. Display employees who are in the salary range 3000 and 5000.
SQL> select * from emp where sal between 3000 and 5000;

IN (set)
Match any of the lists of values

E.g. Display employees whose department no is 20 or 30.
SQL> select * from emp where deptno in (20, 30);

Like condition matches a character pattern.
E.g. Display employees whose name starts with S.
SQL>select * from emp where ename like 'S%';

E.g. Display employees whose name ends with T.
SQL> select *from emp where ename like '%T';

E.g. Display employees whose name contains M as second character.

SQL>select * from emp where ename like '_M%';

Combining Wildcard Characters

The % and _ symbols can be used in any combination with literal characters. You can use the escape identifier to search for the actual % and _ symbols. When you need to have an exact match for the actual % and _ characters, use the ESCAPE option. This option specifies what the escape character is. If you want to search for strings that contain 'SA_', you can use the following SQL statements.

SQL> select empno, ename, job from emp where job like '%SA_%' escape '\'

E.g. Display the names of the employees who are not entitled to get a comm.

SQL>select * from emp where comm is null;

2.2 Logical operators

'AND' Operator: This returns true if both component conditions are true.

E.g. Display the details of clerks in departments 10.
SQL>select * from emp where job = 'CLERK' and deptno=10;

'OR' Operator: This returns true if either of the component condition is true.

Ex: Display the details of employees who are manager or sal>3000.
SQL>select * from emp where job = 'MANAGER' or sal>3000;

'NOT' Operator: This returns true if the following condition is false.

E.g. Display all the details of employees whose job id is not a manager or a president.
SQL>select * from emp where job not in ('MANAGER', 'PRESIDENT');

2.3 Order By Clause

Sorts rows with the ORDER BY clause

Ascending order is default order by comes last in the select statements if Desc is used after the order by clause the rows are displayed in the descending order of that column.

E.g. Display the information of employees in the ascending order of their salary.
SQL>select * from emp order by sal;

E.g. Display the information of employees in descending order of salary.
SQL>select * from emp order by sal desc;

Note:

- If the order by clause is not used the sort order is undefined and the oracle server may not fetch rows in the same order for the same query twice.
- Use the ORDER by clause to display the rows in a specific order.
- We can Sort on multiple columns. The order of ORDER BY list is the order of sort.
- You can order by a column which is not in the select list.

Questions:

1. List all the information about all the employees from emp table.
2. List al employees’ names along with their salaries from emp table.
3. List the names of Analyst and Salesmen.
4. List names of employees who are not managers.
5. List the names of the employees and designation (job) of the employee, who does not report to body. (Manager is null).
6. List the employees who are not eligible for commission.
7. List the employee details not belonging to the department 10 and 20.
8. List the empno, ename, sal in ascending order of salary.
9. List the employee name and hiredate in descending order of hiredate.

3. SQL FUNCTIONS

SQL functions are used to perform calculations on data. To manipulate output from group of rows. It can also format date and numbers for display. It can also be used for modifying individual data items. SQL functions something take arguments and always return a value.

3.1. There are two distinct types of functions

- Single Row Functions
- Multi Row Functions

Single Row Functions operator on single rows only and return one result per row. The different types of single_row functions are

- Character functions
- Number functions
- Date functions
- Conversion Function
- General Function

Note:

The DUAL table is owned by the user 'SYS' and may be accessed by all users. It contain one columns, DUMMY and one row with a value 'X'. The DUAL table is useful when you want to return a value once only – for instance, the value of a constant, pseudo_column or expression that is not derived from a table with 'user' data.

3.2. Character Functions:

UPPER: This returns char, with all letter into uppercase.

SQL> Select upper ('oracle') from dual;

LOWER: This converts the mixed case or uppercase character strings to lower case.

SQL> Select LOWER ('ORACLE') from dual;

INITCAP: Converts the first letter of each word to uppercase and remaining letters to lowercase.

SQL> Select INITCAP ('deccan infotech') from dual;

CONCAT: Joins values together you are limited to two arguments with concat.

SQL> Select CONCAT ('sql', 'functions') from dual;

SUBSTR: This extracts a string of determined length.

SQL> Select SUBSTR ('OracleDBA', 1,6) from dual;

LENGTH: Shows the length of a string as a numeric value.

SQL> Select LENGTH ('Oracle') from dual;

INSTR: Finds numeric position of a named character.

SQL> select INSTR ('Sarfaraz','f') from dual;

LPAD: Pads the character value right_justified.

SQL> Select LPAD(sal, 6, '*') from emp;

RPAD: pads the character value left_justified.

SQL> select RPAD('sal, 6, '*') from emp;

TRIM: Trims heading or trailing characters from a character string.

SQL> select TRIM ('S' FROM 'SSMITHS') from dual;

REPLACE: to replace a set of characters.

SQL> select REPLACE ('JACK AND JUE', 'J', 'BL') from dual;

TRANSLATE: Change a character to a new described character.

SQL> select TRANSLATE ('jack', 'j', 'b') from dual;

3.3 Number Functions

ROUND: Round value to specified decimal.

SQL> select Round(35.823,2), Round(35.823,0), Round(35.823,-1) from dual;

TRUNC: The TRUNC function truncates the column, expression, or value to n decimal places.

SQL> select TRUNC (35.823,2), TRUNC(35.823) from dual;

MOD: The MOD function finds the reminder of value 1 divided by value2.

```
SQL> select ename, sal, mod(sal,2000)
                From emp
                Where job= 'SALESMAN';
```

3.4 Date Function

Date functions operate on Oracle dates. All date functions return a value of DATE datatype except MONTHS BETWEEN which returns a numeric value.

Oracle stores dates in an internal numeric format, representing

Century	Year	Months	
Day	Hours	Minutes	Seconds

The default display/input format for any date is DD-MM-YY. Oracle dates can range between 1st Jan 4712 BC and 31st Dec 4712 AD.

SYSDATE

Is a pseudo-column that returns the current date and time. You can use SYSDATE just as you would use any other column name. For example, you can display the current date by selecting SYSDATE from a table. It is customary to select SYSDATE from dummy table called DUAL.

E.g. To Display the current date:

```
SQL>select sysdate from dual;
```

You would have easily selected SYSDATE FROM EMP, but 14 rows of the same SYSDATE would have been returned, one for every rows of the EMP table. DUAL is preferred because it conveniently returns one row only.

Using Arithmetic Operators

Due to the fact that the date is stored as a number, it is possible to perform calculations with dates using arithmetic operators such as addition and subtraction. You can add and subtract number constants as well as other dates from dates.

The operations you may perform are:

Date+number	add a number of days to a date, producing a date.
Date-number	subtracts number of days from a date, producing a date.
Date-date	subtracts one date from another, producing a number of days.
Date+number/24	add a number of hours to a date producing a date.

```
SQL>select hiredate, hiredate+7, hiredate-7, sysdate-hiredate
            From emp
            Where hiredate like '%jun%';
```

Subtracting SYSDATE from the HIREDATE column of the EMP table returns the number of days since each employee was hired.

Months_Between (Date1, Date2)

Finds the number of months between date1 and date2. The(date1, date2) result can be positive or negative. If date1 is later than date2, the result is positive; if date 1 is earlier than date2, the result is negative.

```
SQL>select monts_between(sysdate, hiredate), months_between('01-jan-84', '05-nov-88')
        From emp;
```

Add_Months(Date, N)
Add n number of calendar months to date 'N' must be an integer and can be negative.

```
SQL>  select hiredate, add_months(hiredate,3), add_months(hiredate,-3)
        from emp where deptno=20;
```

Next_Day(Date1,Char)
Date of the next specified day of the week(char) following date 1, Char may be a number representing a day, or a character.

```
SQL>  select hiredate, next_day(hiredate, 'Friday',) next_day(hiredate, 6)
        from emp
        where deptno=10;
```

Last_Day(Date 1)

```
SQL> Select sysdate, last_date(sysdate), hiredate, last_day(hiredate), last_day('15-feb-88')
        from emp
        where deptno=20;
```

The Round Function Can Be Applied to Dates
ROUND(date1)
Returns date2 with the time set to 12:00AM(midnight). This is useful when comparing dates that may have different times.

ROUND(date1, 'MONTH')
Returns the first to the month containing date1 if date1 is in the first half of the month; otherwise returns the first of the following month.

ROUND(date1, 'YEAR')
Returns the first day of the year containing date1, if date 1 is in the first half of the year, otherwise returns the first of the following year.

```
SQL>select sysdate, round(sysdate,'month'), round(sysdate,'year')
        from dual;
```

TRUNC(date1, 'char')
Finds the date of first day of the month containing in date 1 when char='MONTH'. If char='YEAR', it finds first day of year containing date 1.

```
SQL>select sysdate, trunk(sysdate,'month'), trunk(sysdate, 'year')
        from dual;
```

TRUNC is useful if you want to remove the time portion of the day. The time component of the day is in fact removed by default.

3.5 Conversion Functions

SQL provides a number of function to control date type conversion. These conversion functions convert a value from one datatype to another.

TO_CHAR(number | date, ['fmt'])	Converts number of date to character format fmt.
TO_NUMBER(char)	Converts char, which contains a number to a NUMBER.
TO_DATE('char', 'fmt')	Converts the char value representing date, into a date value according to fmt specified. If fmt is omitted, format is DD-MM-YY.
TO_CHAR	The TO_CHAR function is frequently used to change a date format from the default to an alternative display format.
TO_CHAR (date, 'date picture')	To convert the current date from the default format (DD-MON-YY) to A new 'date picture'.

```
SQL> selected to_char (sysdate,'day,ddth month yyyy')
        from dual;
```

Note:

The 'date picture', which must be embedded within single quotes, can include any of the format listed below. The column and 'date picture' must be separated by comma.

Day and Month in the output are automatically padded with blanks to a length of 9 to remove the blank padding use the FM (Fill Mode) prefix:

```
SQL>select to_char(sysdate,'fmday, ddth month yyyy')
        from dual;
```

FM can be used to suppress leading zeroes for the ddth format, e.g. 05TH is changed to 5th. The case in which the 'date picture' is entered is the case in which it will be displayed.

TO_CHAR can also be used to exact the time of day only, and display it in a specified format. To display the time of the day:

```
SQL>select to_char(sysdate, 'hh:mi:ss')
        from dual;
```

The TO_CHAR function is also used to convert a value of NUMBER data type to a value of CHAR datatype.

TO_CHAR (number, 'number picture')

```
SQL>select to_char(sal,'$9,999')
        from emp;
```

If the 'date picture' is omitted, the date is converted to a char value in ORACLE's default date format 'DD-MM-YY'. If the 'number picture' is not specified, the number is converted to a char value. Also note that format models do not affect the actual internal representation of the column value. They only affect how the column value is displayed when retrieved with a SELECT statements.

The RR Date Format Element
If you use the RR date format element instead of YY, the century of the return value varies according to the specified two-digit year and the last two digits of the current year. The table summarizes the behavior of the RR element.

To_Number
It is used to transform a number stored as a character to number datatype:

```
SQL> select empno, ename, job, sal
        from emp
        where sal>to_number ('1500');
```

To_Date
To show all employees hired on June 4, 1984 (non-default format) we can use the TO_DATE function:

```
SQL>select empno, ename, hiredate
        from emp
        where hiredate=to_date('june 4, 1984', 'month dd, yyyy');
```

The constant is converted to a date and then compared to the HIREDATE value.
To TO_DATE function is frequently used to supply a value to ORACLE expects to be passed a date with the default date format of DD-MM-YY. If you do not want to use the default date format, you must use the TO_DATE: function and the appropriate alternative format mask.

To enter a row into the EMP table with a non-standard date picture enter:

```
SQL>insert into emp (empno, deptno, hiredate)
                values(7777, 20, to_date('19/08/90', 'dd/mm/yy');

SQL> UPDATE emp SET hiredate = TO_DATE('1998 05 20', 'YYYY MM DD')
        WHERE ename='SMITH';
```

3.6 General Functions
The following are the some of the function supported by oracle:

Uid
This function returns the integer value corresponding to the user currently logged in.

```
SQL>select uid
        from dual;
```

User
This function returns the login's user name, which is in varchar2 datatype.

```
SQL>select user
        from dual;
```

Nvl
This function is used in case where we want to consider Null values as zero.

```
SQL>select ename, nvl(comm.,0)
        from emp;
```

Vsize
The function returns the number of bytes in the expression if expression is null it returns null.

```
SQL>select vsize('hello')
        from dual;
```

Case
CASE expression let you use IF-THEN-ELSE logic in SQL statements without having to invoke procedures.

```
SQL>select ename job, sal,
                        case job when 'CLERK' then 1.10*sal
                        when 'MANAGER' then 1.15*sal
                        when 'SALESMAN' then 1.20*sal
                        else sal end "REVISED SALARY"
                from emp;
```

In this query Oracle searches for the first WHEN..THEN pair for which expr is equal to comparison_expr and return return_expr. If none of when..then pairs meet this condition and an else clause exists, then Oracle returns else_expr. Other wise oracle returns null.

DECODE function
The decode function decodes an expression in a way similar to the IF-THEN-ELSE logic used in various language. The DECODE function decodes expression after comparing it to each search value. If the expression is same as search, result is returned.

```
SQL>select ename job, sal,
  DECODE (job,'CLERK', 1.10*sal, 'MANAGER', 1.15*sal, 'SALESMAN', 1.20*sal,sal)
   "revised salary"
   from emp;
```

3.7 Group function or Multiple Row function

A group function returns a result based on a group of rows. Some of these are just purely mathematical functions.

The group functions operate on sets of rows to give one result per group. These sets may be the whole table or the table split into groups. The group function supported by oracle are sum, avg, count, min, max. You can use avg and sum against columns that can store numeric data. You can use max and min for any datatype.

Sum
The sum function can be used to obtain the sum of a range of values of a record set.

```
SQL>select sum(sal) “Total Salary”)
          from emp;
```

Avg
The avg function will return the average of values of the column specified in the argument of the column.

```
SQL>select avg(sal)
          from emp;
```

Min
This function will give the least of all value of the column present in the argument.

```
SQL>select min(sal) “Minimum Salary”
          from emp;
```

Max
To perform an operation which gives the maximum of a set of values the max function can be made use of :

```
SQL>select max(sal)
          from emp;
```

Count
It counts all rows, inclusive of duplicates and nulls.

```
SQL>select count(*)
          from emp;
```

3.8 Group by Clause

You can use the GROUP BY CLAUSE to divide the rows in a table into groups. You can then use the group functions to return summary information for each group.

E.g.: Display sum of salaries department wise.

```
SQL>select deptno, sum(sal)
          from emp
          group by deptno;
```

E.g.: Display no of employees under each job category.

```
SQL>select job, count(*)
          from emp
          group by job;
```

E.g.: Display the number of employees in department 30.

```
SQL>select deptno, count(*)
        from emp
        where deptno = 30
        Group by deptno;
```

Group by clause displays one line of data for each department retrieved in the WHERE clause, and COUNT(*) displays the number of employees in each department group displayed.

3.9 Use the HAVING clause to restrict groups:

You can use the GROUP BY clause without using a group function in the select list if you restrict rows based on the result of a group function, then you must have a GROUP BY clause as well as the HAVING clause.

- Rows are grouped
- The group function is applied
- Group matching the having clause are displayed.
- The WHERE clause cannot be used to restrict groups.

You use the HAVING clause to specify which group are to be displayed. Therefore your further Restrict the groups on the basis aggregate information.

Syntax

```
        SELECT column group_function
        FROM table
        [where condition]
        [group by group_by_expression]
        [having group_condition]
        [order by column];
```

E.g.: Display the total salary of department 20.

```
SQL>select deptno,sum(sal)
        from emp group by deptno
        Having deptno=20;
```

E.g.: Display the department number and average salary for those departments that have an average salary more than 2000.

```
SQL> select deptno, avg(sal)
        from emp
        group by deptno having avg(sal)>2000.
```

Note:
If you restrict rows based on the result of a group function, you must have a GROUP BY CLAUSE as well as the HAVING clause.

QUESTIONS

1. List the employees name ending with an 'S'.
2. List the employees name having 'I' as the second character.
3. List the number of jobs available in the emp table.
4. List the number of employees working with the company.
5. List the number of employees working with the company.
6. List the department number and number of employees in each department.
7. List the total salary, maximum and minimum salary and the average salary of employees job Wise, for department number 20 only.
8. List the average salary for all departments employing more than five people.
9. List the maximum salary of employee working as salesman.
10. List the average salary and number of employees working in the department 20.
11. What is the different between group and order by?
12. Display the name in the lowercase, job in the initial capitalization for all managers.
13. Display the value 45.923 rounded to the hundredth, no, and ten decimal places.
14. Calculate the remainder of the ratio of salary to commission for all employees whose salary is more than 1000.
15. For Employees in department 30, display the ename and number of weeks employed.
16. Display the head of the company, who has no manager, display that there is 'no manager' for that name.
17. Display the job and total salary for each job category with a total salary exceeding 3000. and sort list by total salary.
18. Display the number of managers without listing them.
19. Write a query to display the minimum and maximum salary for each job type ordered alphabetically.

4. DISPLAYING DATA FROM MULTIPLE TABLES

4.1 Set operators:

Set operators combine the result of two component queries into a single result. Queries containing set operator called compound queries. They combine the result of two or more select statements into the result. A query may therefore consist of two or more SQL statements linked by one or more set operators. Set operators are often called vertical joins, because the join is not according to rows between two tables, but columns.

You can combine multiple queries using the set operators UNION, UNION ALL, INTERSECT, and MINUS. All set operators have equal precedence. If a SQL statement contains multiple set operators, Oracle evaluates them from the left to right if no parentheses explicitly specify another order.

The corresponding expression in the select lists of the component queries of a compound query must match in number and datatype. If component queries select character data, the datatype of the return values are determined as follows:

- If both queries select values of datatype CHAR, the returned values have datatype CHAR.
- If either or both of the queries select values of datatype VARCHAR2. the returned values have datatype VARCHAR2.

Restriction on set operators:

- The set operators are not valid on column of type BLOB, CLOB, BFILE, varray, or nested table.
- The UNION, INTERSECT, and MINUS operators are not valid on LONG column
- To reference a column, you must use an alias to name the column
- You cannot also specify the for_update_clause with these set operators.

UNION
All rows selected by either query.

```
SQL>select job from emp where deptno=10
        Union
        Select job from emp where deptno=30;
```

UNION ALL
All rows selected by either query, including all duplicates.

```
SQL>select job form emp where deptno=10
        Union all
        Select job from emp where depnto=30;
```

INTERSECT
All distinct rows selected by both queries.

```
SQL>select job from emp where deptno=10
        Intersect
        Select job from emp where deptno=30;
```

MINUS
All distinct rows select by the first query but not the second.

```
SQL> select job from emp where deptno=10
        Minus
        Select job from emp where deptno=30;
```

4.2 Relating data through join concept

Join is a query that combines rows from two or more tables or views. Oracle performs a join whenever multiple tables appear in the queries FROM clause. The query's select list can select any columns from any of these tables. If any two of these tables have a column name in common, you must qualify all references to these columns throughout the query with table names to avoid ambiguity.

Join condition:
Most join queries contain WHERE clause condition that compare two columns, each from a different table. Such a condition is called a join condition. To execute a join Oracle combines pairs of rows, each containing one row from each table for which the join condition evaluates to TRUE. The columns in the join conditions need not also appear in the select list. To execute a join of three or more tables. Oracle first joins two of the tables based on the join conditions comparing their columns and then joins the result to another table based on join conditions containing columns of the joined tables and the new table. Oracle continues this process until all tables are joined into the result.

There are three different types of joins;

4.3 Simple Join
It is the most common type of join. It retrieves from two tables having a common column and is further classified into **equi-join** and **non equi-join.**

A join, which is based on equalities, is called an **equi-join.**

E.g. Display all employees name and their department name in desc order of the department name.

```
SQL>select e.ename, d.dname
        From emp e, dept d
        Where e.deptno=d.deptno and sal> 1500
```

E.g.: Display all employees working in NEWYORK.

```
SQL>select emp.ename, emp.sal, dept.loc
        from emp, dept
        where emp.deptno=dept.depnto and loc= 'NEWYORK';
```

E.g.: Display the employee name, department number, and department name

```
SQL>select emp.ename, dept.dname, emp.deptno
        from emp, dept
        Where emp.deptno=dept.depnto;
```

4.4 Using Table Aliases

Simplify queries by using table aliases. Improve performance by using table prefixes. Qualifying column names with table names can be very time consuming, particularly if table names are lengthy. Table aliases help to keep SQL code smaller, therefore using less memory.

4.5 Non Equijoins

A non equi-join specifies the relationships between columns belonging to different tables by making use of the relational operator (>, <, <=, >=, <>) other than equal to (=).

The relationship between the EMP and SALGRADE tables is a non-equijoin, in that no column in EMP corresponds directly to a column in SALGRADE. The relationship is obtained using an operator other than equal (=).

E.g.: Display the employee's salary grade.

```
SQL>select e.ename, e.sal, s.grade
        from emp e, salgrade s
        where sal between losal and hisal;
```

E.g.: Display all employees on grade 3.

```
SQL>select ename, sal, job, grade
        from emp, salgrade
        where grade=3;
```

4.6 Self Joins

A self join is a join of a table to itself. This tables appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition. To perform a self join, Oracle combines and returns rows of the table that satisfy the join condition.

To find the name of each employee's manager, you need to join the employees table to itself, or perform a selfjoin.

E.g.: Display the name and their manager name.

```
SQL> select worker.ename||' works for '||manager.ename
        from emp worker, emp manager
        where worker.mgr=manager.empno;
```

E.g.: Display employees whose salary is greater than their boss salary

```
SQL> Select el.ename
        from emp el, emp e2
        Where el.mgr=e2.empno and e1.sal>e2.sal;
```

4.7 Outer Join

The outer join extends the result of a simple join. An outer join returns all the rows returned by simple join as well as those rows from one table that do not match any row from the other table this cannot be done with simple join.

The symbol(+) represents outer join The outer join operator can appear on only on one side of the expression – the side that has information missing. It returns those rows from one table that have no direct match in the other table.

E.g.: Display all information along with the departments that do not have employees.

```
SQL>select ename, sal, job, dname, dept.deptno
        From emp.deptno(+)=dept.deptno;
```

4.8 Cartesian Products

If two tables in a join query have no join condition, Oracle returns their Cartesian product. Oracle combines each row of one table with each row of the other. A Cartesian product always generates many rows and is rarely useful for example, the Cartesian product of two Tables each with 100 rows has 10000 rows. Always include a join condition unless you specifically need a Cartesian product. If a query joins three or more tables and you do not specify a join condition for a specific pair, the optimizer may choose a join order that avoids producing an intermediate Cartesian product.

Questions

1. What is the basic different between a join and a union?
2. Explain an outer join
3. List the employees number, name, departments numbers and the department name;
4. Display the ename and dname of employees and their departments in desc order of deptno.
5. Display the list of employees working in each department. Display the department information.
6. Even if no employee belongs to that department.
7. List all employees who joined the company before their manger.
8. List the jobs common to department 20 and 30.
9. List the jobs unique to department 20.
10. Display the names of employees and their respective manger.

5. SUB QUERIES

A subquery is a form of select statement which appears inside another SQL statement. A statement containing a subquery is called a parent statement. Subquery are used to retrieve data from table that depend on the values in the table itself.

5.1 Use sub queries for the following purposes:

- To define the set of rows to be inserted into the target table of an INSERT or CREATE TABLE statement.
- To Define the set of rows to be included in a view or materialized view in a CREATE VIEW or CREATE MATERIALIZED VIEW statement.
- To define one or more values to be assigned to existing rows in an UPDATE statement.
- To provide values for conditions in a WHERE clause, HAVING clause, a START WITH clause of SELECT, UPDATE, and DELETE statement.
- To define a table to be operated on by a containing query.

You do this by placing the subquery in the FROM clause of the containing query as you would a table name. you may use subqueries in place of tables in this way as well in INSERT, UPDATE, and DELETE statement.

Subqueries so used can employ correlation variable, but only those defined within the subquery itself, not outer references. Outer references ("left-correlated subqueries") are allowed only in the FROM clause of a SELECT statement.

5.2 Types of Subqueries

Single-row subqueries: Queries that return one row from the inner SELECT statement.
Multiple-row subquery: Queries that return more than one row from the inner SELECT statement.

Note: Always enclosed subquery within parenthesis. Subquery will be evaluated first followed by the main query you can place the subquery in a number of SQL clauses including the where clause, The Having clause, the from clause.

5.3Executing Single-Row Subqueries

E.g.: Display the employees whose job is same as that of 7788.

```
SQL> select ename,job,sal from emp
        Where job=(select job from emp where empno=7788)
```

E.g.: Display all employees who have the same job as blake.

```
SQL> select ename, job from emp
        Where job=(select job from emp where ename='BLAKE');
```

Using group functions in a subquery you can display data from a main query by using a group function in a subquery to return a single row.

E.g.: Display the employees who earns the minimum salary in the company.
SQL> select * from emp where sal=(select min(sal) from emp);

Note: The inner query returns a value that is used by the outer query or the main query.

Using Having clause with subqueries you can use subqueries not only in the where caluse, but also in the having clause. The Oracle server executes the subquery and the results are returned into the having clause of the main query.

E.g.: Find the job with the lowest average salary.

```
SQL>select job, avg(sal) from emp group by job
        Having avg(sal) = (select min(avg(sal)) from emp group by job);
```

5.4 Multiple-Row Subqueries

Subqueries that return more than one row are called multiple-row subqueries, you use multiple row operator, instead of a single row operator, with a multiple-row subquery. The multiple row operators expect one or more values.
The multiple row comparison operators are IN, ANY, ALL

E.g.: Display employees who earn the lowest salary in each department.

```
SQL>select ename, sal, deptno from emp
        Where sal in(select min(sal) from emp group by deptno);
```

E.g.: Display the employees who are not managers and whose salary is less than that of any Manager.

```
SQL> select empno, ename, job, sal
        from emp
        where sal<any(select sal from emp where job='MANAGER');
```

5.5 Correlated Subqueries

A correlated subquery is a nested subquery, which is executed once for each 'candidate row' considered, buy the main query and which on execution uses a value from a column in the outer query. This causes the correlated subquery to be processed in different way from the ordinary nested subquery.

A correlated subquery is identified by the use of an outer query's column in the inner query's predicate clause. With a normal nested subquery, the inner select runs first and it executes once, returning values to be used by the main query. A correleated subquery, on the other hand, executes once for each row(candidate row) considered by the outer query. The inner query is driven by the outer query.

Steps to execute a correlated sub-query

1. Get candidate row(fetched by outer query).
2. Execute inner query using candidate row's value.
3. Use values(s) resulting from the inner query to qualify or disqualify candidate.
4. Repeat until to candidate row remains.

Although the correlated subquery executes repeatedly, once for each in the main query, there is no suggestion that correlated subqueries are less efficient than ordinary non-correlated subqueries.

We can use a correlated subquery to find employees who earn a salary greater than the average salary for their department.

```
SQL>select empno, ename, sal, deptno
        from emp
        where sal>(select avg(sal) from emp where deptno=e.deptno)order by deptno;
```

We can see immediately that this is a correlated query since we have a column from the outer SELECT in the WHERE clause of the inner SELECT.

Note that the alias is necessary only to avoid ambiguity in column names.

Now, let us analyze the above example using the EMP table:

The Main Query

- Select first candidate row in department 20 earning 800.
- EMP in FROM clause has alias E which qualifies DEPTNO column references in inner query's WHERE clause.
- WHERE clause compares 800 against value returned by inner query.

The inner Query

- Computes AVG(SAL) for employee's department.
- WHERE department value is candidate's department (E DEPTNO) value passed into inner query from outer query's DEPTNO column.
- Candidate row does not meet condition. So, discard.
- Repeat from step 1 for next candidate row.

The selection of candidate rows continues with those meeting the condition appearing in the query result. A correlated subquery is signaled by a column name, a table or table alias in the WHERE clause that refers to the values of a column in each repeatedly for each candidate row in the main query.

Operators

When you are nesting select statement, the logical operators are all valid as well as ANY and ALL. In addition the EXIST operators may be used., EXIT OPERATOR.

The EXISTS operator is frequently used with correlated subqueries. It tests whether a value is there (NOT EXISTS) ensures that something is not there. If the value exists it returns TRUE; if it does not exist FALSE is flagged.

To find employees who have at least one person reporting to them, enter.

```
SQL>select empno, ename, job, deptno from emp
        Where exists (select empno from emp where emp mgr-------------)
        Order by empno;
```

E.g.: Display employees who earns a salary greater than the average for their departments.

```
SQL> Select ename, sal, deptno from emp
        Where sal > (select avg(sal) from emp where deptno=e.deptno) order by deptno;
```

E.g.: Find employee who have at least one person reporting to them.

```
SQL>select empno, ename, job, deptno from emp
        Where empno in (select mgr from emp where mgr is not null);
```

Questions

1. Display the details of employees who work in new york.
2. List the employees belonging to the department of MILLER .
3. List the name of the employees drawing the highest salary.
4. List employee details who earn salary greater than the average salary for their department.
5. List all employees who have at least one person reporting to them.
6. List all the employees details who do not manage any one.
7. List the employee details of those employees whose salary is greter than any of the mangers.
8. List the employee names whose salary is greater than the highest salary of all employees belonging to departmentno. 20.
9. List the details of the employee earning more than the highest paid MANAGER.
10. Display all the department that have an average salary is greater than that of department 20.

6. Data Manipulation Language (DML)

6.1 Datatypes

Each value manipulated by Oracle has a datatype. A values datatype associated a fixed set of properties with the value. These properties cause Oracle to treat values of one datatype differently from values of another. For example you can add values of NUMBER datatypes, but not values of RAW datatype.

When you create a table, you must specify a datatype for each of its columns. When you create a procedure or stored function, you must specify a datatype for each of its arguments. These datatypes define the domain of values that each column can contain or each argument can have. For example, DATE columns cannot accept the value February 29 (Except for a leap year) or the values 2 or 'SHOE'. Each value subsequently placed in a column assumes the column's datatype. For example, if you insert '01-JAN-1998' into a DATE column, Oracle treats the '01-JAN-98' character string as a DATE value after verifying that it translates to a valid date. Oracle provides a number of built-in datatypes as well as several categories for user-defined types.

Character Datatypes:
Character datatypes store character (Alphanumeric) data, which are words and free-form text, in the database character set or national character set. They are less restrictive than other datatypes and consequently have fewer properties. For example, character columns can store all alphanumeric values, but NUMBER columns can store only numeric values.

VARCHAR2 datatype;
The VARCHAR2 datatype specifies a variable-length character string. When you create a VARCHAR2 Column you supply the maximum number of bytes or character of data that it can hold. Oracle subsequently stores each value in the column exactly as you specify it, provided the value does not exceed the column's maximum length. If you try to insert a value that exceeds the specified length, Oracle returns an error. You must specify a maximum length for a VARCHAR2 column. This maximum must be at least 1 byte, although the actual length of the string stored in permitted to be zero. Oracle treats zero-length strings as nulls. You can use the CHAR qualifier, for example VARCHAR2(10 CHAR), to give the maximum length in characters instead of bytes. The maximum length of VARCHAR2 data is 4000 bytes. Oracle compares VARCHAR 2 values nonpadded comparison semantics.

NUMBER Datatypes;
The NUMBER datatypes stores zero, positive, and negative fixed and floating-point numbers with magnitudes between 1.0 x 10^{-130} and 99…9 x 10^{125} (38 nines followed by 88 zeroes) with 38 digits of precision. If you specify an arithmetic expression whose value has a magnitude greater than or equal to 1.0 x 10^{126} Oracle returns an error.

Specify a fixed-point number using the following form:
NUMBER (p,s),

Where:

- p is the precision, or the total number of digits. Oracle guarantees the portability of numbers with precision ranging from 1 to 38.
- s is the scale, or the number of digits to the right of the decimal point. The scale can range from -84 to 127.

Specify an integer using the following form:
NUMBER(p) This represents a fixed-point number with precision p and scale 0 and is equivalent to NUMBER (p,0).

Long datatype ;
LONG columns store variable-length character strings containing up to 2 gigabytes, or 2^{31}-1 bytes. LONG columns have many of the characteristic of VARCHAR2 columns. You can use LONG column to store long txt strings. The length of LONG values may be limited by the memory available on you computer.

Raw datatype:
Raw data type is used to store byte-oriented data like binary data or byte strings and the maximum size of this datatype is 2000 bytes.

Long Raw Datatype:
Long raw datatype is used to store binary data of variable length, which can have a maximum size of 2GB.

Date datatype:
The DATE datatype stores date and time information. Although date and time information can be represented in both character and number datatypes, the DATE datatype has special associated properties. For each DATE value, Oracle stores the following information: Century, year, month, date, hour, minute, and second.

You can specify a date value as a literal, or you can convert a character or numeric value to a date value with the TO_DATE function. To specify a date as literal, you must use the Gregorian calendar. You can specify an ANSI date literal, as shown in this example:
DATE ‘1998-12-25’

In addition, to the above Oracle supports:

CLOB
BLOB
BFILE

CLOB stores character objects with single byte characters. A table can have multiple columns with CLOB as its data type.

BLOB can store large binary objects as graphics, video clips and sound files. A table can have Multiple columns with BLOB as its datatype.
Bfile column stores file pointers to LOBs managed by the file system external to the database a BFILE column may contain filenames for photos stored on the CD-ROM.

Large Object (LOB) Datatypes ;
The built-in LOB datatypes BLOB, CLOB and NCLOB (stored internally) and BFILE (stored externally), can store large and unstructured data such as text, image, video, and spatial data up to 4 gigabytes in size.

When creating a table, you can optionally specify different tablespace and storage characteristic for LOB columns or LOB object attributes from those specified for the table.

LOB columns contain LOB locators that can refer to out-of-line or in-line LOB values. Selecting a LOB from a table actually returns the LOB's locator and not the entire LOB value. The DBMS_LOB package and Oracle Call interface (OCI) operations on LOBs are performed through these locators.

LOBs are similar to LONG and LONG RAW types, but differ in the following ways:

- LOBs can be attributes of a user-defined datatype (object).
- The LOB locator is stored in the table column either with or without the actual LOB value. BLOB, NCLOB, and CLOB values can be stored in separate tablespace. BFILE data is stored in an external file on the server.
- When you access a LOB column, the locator is returned.
- A LOB can be up to 4 gigabytes in size. BFILE maximum size is operating system dependent, but cannot exceed 4 gigabytes.
- LOB's permit efficient, random, piece-wise access to and manipulation of data.
- You can define more than one LOB column in a table.
- With the exception of NCLOB, you can define one or more LOB attributes in an object.
- You can declare LOB blind variables.
- You can select LOB columns and LOB attributes.
- You can insert a new row or update an existing row that contains one or more LOB columns and /or an object with one or more LOB attributes. (You can set the internal LOB value to NULL, empty, or replace the entire LOB with data. You can set the BFILE to NULL or make it point to a different file).
- You can delete a row contain a LOB column or LOB attribute and thereby also delete the LOB value. Note that for BFILEs the actual operating system file is note deleted.

You can access and populate rows of an internal LOB column (a LOB column stored in the database) simply by issuing an INSERT or UPDATE statement. However, to access and populate a LOB attribute that is part of an object type, you must first initialize the LOB attribute using the EMPTY_CLOB or EMPTY_BLOB function. You can then select the empty LOB attribute and populate it using the DBMS_LOB package or some other appropriate interface.

BFILE datatypes ;
The BFILE datatype enables access to binary file LOBs that are stored in file system outside. Oracle database a BFILE column or attribute stores a BFILE locator, which serves as a pointer to a binary file on the server's file system. The locator maintains the directory alias and the filename.

BLOB Datatype ;
The BLOB datatype stores unstructured binary large objects. BLOBs can be thought of as bit streams with no character set semantics. BLOBs can store up to 4 gigabytes of binary data.

BLOBs have full transaction support. Changes made through SQL, the DBMS_LOB package, or the OCI participate fullt in the transaction. BLOB value manipulation can be committed and rolled back. However, you cannot save a BLOB locator in a PL/SQL or OCI variable in one transaction and then use it in another transaction or session.

CLOB Datatype;
The CLOB datatype stores single-byte and multi byte character data. Both fixed-width and variable-width character sets are supported and both use the CHAR database character set. CLOBs can store up to 4 gigabytes of character data.

CLOBs have full transaction support. Changes made through SQL, the DBMS_LOB package, or the OCI participate fully in the transaction. CLOB value manipulations can be committed and rolled back. However, you cannot save a CLOB locator in PL/SQL or OCI variable in one transaction and then use it in another transaction or session.

Note: Currently varchar is equivalent to varchar2 datatype.

6.2 Data Manipulation Language

Data manipulation language (DML) is a core part of SQL. When you want to add, update, or delete data in the database you execute a DML statement. A collection of DML statement that have not yet been permanent is called a transaction, or a logical unit of work. Data manipulation commands are the most frequently used SQL commands.

Insert command
The insert command is used to add one or more rows to a table. While using this command the values are separated by commas and the datatypes varchar2, char, long are enclosed in single quotes.

The values must be entered in the same order as they are defined in the table.

Syntax: Insert into <table_name> values (a list of values);

Eg: Insert into dept values(50,’IT’,’PUNE’);

If data has to be written into only specific columns this too is possible with slight variations to the general syntax:

Syntax: Insert into <tablename (col_name)>values (list of values);

```
Eg: Insert into emp(empno, ename, job, sal,hiredate)
        Values(101,’Sarfaraz’,’DBA’,4000,sysdate);
```

Copying Rows from another table
You can use the insert statement to add rows to a table where the values are derived from the existing table. In place of values clause, you can use a subquery returns rows into the Table.

Syntax:
Insert into table [column (, column)] subquery;

Eg: Insert into dept2 (select * from emp);

Update Command
The update command consists of a 'set' clause and an optional 'where' clause to reflect changes to the existing data update command is used. With the update command we can update rows in the table. A single column may be updated or more than one column could be updated. Specific rows could be updated based on a specific condition.

Syntax
```
Upate table_name
set field_1=value,
field_2=value
where condition;
```

```
Eg:     Update dept
        Set dname = 'ACCOUNTS'
        Where deptno=50;
```

Note: If the where condition is commited, the entire table will updated.

```
Eg:
        Update emp
        Set job='DBA',
            Sal= 40000
        Where empno=101;
Eg:
        Update emp
        Set sal = (select sal from emp where empno=101)
        Where empno=7902;
```

Delete Command
After inserting rows in a table we can also delete them if required. The delete command consists of a 'from' clause followed by the optional 'Where' clause.

Syntax:
Delete from <table_name>where conditions;

```
Eg:
        Delete from emp;
         - - All the records will be deleted.
```

```
Eg:     Delete from emp
        Where job = 'DBA';
```

Merge command
SQL has been extended to include the merge statement. Using this statement, you can update or insert a row conditionally into a table, thus avoiding multiple UPDATE statements. The decision whether To update or insert into the target table is based on a condition in the ON clause. Since the MERGE command combines the INSERT and UPDATE commands. You need both INSERT and UPDATE privilages on the target table and the SELECT privilege on the source table. The MERGE statement is deterministic. You cannot update the same row of the target table multiple times in the same MERGE statement. The MERGE statement is suitable in a number of data warehousing applications. For example, in a data Warehousing application, you may need to work with data coming from multiple sources, some of which may be duplicates. With the MERGE statement, you can conditionally add or modify rows.

```
MERGE INTO table_name table_alias
USING (table/view/sub_query) alias
        ON(join condition)
        WHEN MATCHED THEN
                UPDATE SET
                Col1 = col_vall,
                Col2 = col2_val
        WHEN NOT MATCHED THEN
                INSERT (column_list)
                VALUES (column_values);
```

Example of Merging Rows :

```
MERGE INTO copy_emp c
        USING emp e
        ON (c.empno=e.empno)
WHEN MATCHED THEN
        UPDATE SET
        c.ename=e.ename,
        c.job=e.job,
        c.mgr=e.mgr,
        c.sal=e.sal,
        c.comm=e.comm,
        c.deptno=e.deptno
WHEN NOT MATCHED THEN
        INSERT VALUES (e.empno, e.ename, e.job, e.comm, e.mgr, e.deptno);
```

The example shown matches the empno in the copy emp table to the empno in Employee table. If a match is found, the row in the copy_emp table is updated to match the row in the employees table. If the row is not found, it is inserted into the copy_emp table.

6.3 Transaction Control Language

A transaction is a logical unit of work. All Changes made to the database can be referred to as a transaction maintaining security and integrity of a database is the most important factor in judging the success of a system. This integrity can be applied to different degrees of severity. An integrity Constraints is a mechanism used by Oracle to prevent invalid data entry into the table. In other words constraints are used for enforcing rules that the column in a table have to confirm with. Commit and rollback controls the transaction with the database.

Commit

This command is used to end a transaction. Only with the help of the commit command, transaction changes can be made permanent to the database.
This command also erases all savepoints in the transaction thus releasing the transaction locks. COMMIT WORK; (OR) COMMIT;

Advantages of COMMIT and ROLLBACK

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations

Savepoint

Savepoints are like markers to divide a very lengthy transaction to smaller ones. They are used to identify a point in a transaction to which we can latter rollback. Thus savepoints is used in conjunction with rollback portion of current transaction.
All savepoints marked after the savepoint to which you roll back are erased. However, the savepoint to which you roll back is not erased. For example, if you mark savepoints A,B,C, and D in that order, Then roll back to savepoint B, only savepoints C and D are erased.
An implicit savepoint is marked before executing an INSERT, UPDATE, or DELETE statement. If the statement fails, a rollback to the implicit savepoint is done. Normally, just the failed SQL statement is rolled back, not the whole transaction. However, if the statement raises an unhandled exception, the host environment determines what is rolled back.

In SQL, the FORCE clause manually rolls back an in-doubt distributed transaction. However, PL/SQL does not support this clause. For example, the following statement is illegal:
Savepoint savepoint_id;

Rollback

A rollback command is used to under the work done in the current transaction. We can either rollback The entire transaction so that all changes madder by sql statement are undone, or rollback a transaction to a savepoint. So that the SQL statements after the savepoint are rolled back.

SQL>Rollback work; or
Rollback;

Rollback to a particular stage in a transaction, a savepoint
Rollback to savepoint save_pt;
Where save_pt is the savepoint.

7. DATA DEFINITION LANGUAGE (DDL)

Data Definition Language is used to create an object, alter the structure of an object and also to drop the object created.

Table Definition: A table is a unit of storage that hold data in the form of rows and columns.

Clause table commands.

A table is the basic unit of data storage in an Oracle database. The tables of a database hold all of the user-accessible data.

Table data is stored in rows and columns. Every table is defined with a table name and set of columns. Each column is given a column name, a datatype (such a CHAR, DATE, or NUMBER), and a width (which could be predetermined by the datatype, as in DATE) or scale and precision (for the NUMBER datatype only). Once a table is created, valid rows of data can be inserted into it. The table's rows can be queried, deleted, or updated.

Tables are created with no data unless a query is specified. You can add rows to a table with the INSERT statement. After creating a table, you can define additional columns, partitions, and integrity constraints with the ADD clause of the ALTER TABLE statement. You can change the definition of an existing column or partition with the MODIFY clause of the ALTER TABLE statement.

Syntax:

Create table<table_name> (column definition1, column definition2….);

The table name should adhere to the following norm while naming a table:
- The first letter should be an alphabet.
- Oracle reserved words cannot be used as a table name.
- Maximum length for a table name is 30 characters.
- Two different tables should not have same name.
- Underscore, numerals and letter are allowed but not blanks space and single quotes.

```
SQL> create table student
        (id number,
         name varchar2(10),
        course varchar2(10)
        );
```

Alter TABLE Command:
'Alter Table' Command can be use to perform several different operations, as listed below:

To Add a Column:
```
SQL> Alter table student
        Add (location varchar2(10));
```

To Rename a Column:

```
SQL> Alter table student
        Rename column location to branch;
```

To Change The Data Type of a Column:

```
SQL> Alter Table student
        Modify (name varchar2(20));
```

To Drop a Column:

```
SQL> Alter table student
        Drop column branch;
```

The SET UNUSED option

The SET UNUSED option marks one or more columns as unused so that they an be dropped when the demand on system resources is lower. This is a feature available in Oracle8i and later. Specifying this clause does not actually remove the target columns form each row in the table(that is, it does not restore the disk space used by these columns). Therefore, the response time is faster than if you executed the drop clause. Unused columns are treated s if they were dropped, even though their column data remains in the tables row. After a column has been marked as unused, you have no access to that column. A select * query will not retrieve data from unused columns. In addition, the names and types of column marked unused will not be displayed during a DESCRIBE, and you can add toi the table a new column with the same name as an unused column. SET UNUSED information is stored in the USER_UNUSED_COL_TABS dictionary view.

Syntax:

```
ALTER TABLE table name SET UNUSED COLUMN column;
```

```
Eg: SQL>Alter table student
        Set unused course;
```

The DROP UNUSED COLUMNS option;

DROP UNUSED COLUMNS removes from the table all columns currently marked as ununsed. You can use this statement when you want to reclaim the extra disk space from unused columns in the table.

If the table contains no unused columns, the statement returns no errors.

Syntax:

```
ALTER TABLE table name  DROP UNUSED COLUMNS:
```

```
Eg: SQL>Alter table student
        Drop unused columns;
```

If there is no further use of records stored in a table and the structure has to be retained then the records alone can be deleted.

Syntax:
Truncate table<table_name>

SQL> Truncate Table Student;

Command to Drop a table

In order to drop a table, we can use the drop table command supported by SQL.
Use the DROP TABLE statement to remove a table or an object table and all its data from the database.

Syntax:
Drop table<table_name>

SQL> Drop Table Student;

Note:
When a table is dropped, associated objects like indexes, triggers are also dropped.
Integrity Constraints on the table are also drop.
All the other depended object like views, synonyms, procedures, etc exist but are marked invalid.

Adding Comments to a emp table
SQL> Comment on table emp is 'employee information'

Remove a comment from a column

SQL>comment on column emp is ' ';

7.1 Integrity Constraints

An integrity constraint is a declarative way to define a business rule for a column of a table. An integrity constraint is a statement about a table's data that is always true and that follows these rules:

- If an integrity constraint is created for a table and some existing table data does not satisfy the constraints, the constraint cannot be enforced.
- After a constraint is defined, if any of the results of a DML statement violate the integrity constraint, the statement is rolled back and an error is returned.

Integrity constraints are defined with a table and are stored as part of the table's definition, centrally in the database's data dictionary, so that all database application must adhere to the same set of rules. If a rule changes, it need only be changed once at the database level and not many times for each application. All data integrity constraints should be enforced by the database server or the application software.
Following are the Five Types of integrity Constraints with their characteristics:

1> Not Null Constraint: A Columns defined as NOT NULL constraint cannot constrain NULL Values. i.e. for a records to be inserted in the table, value for this columns is MUST.

2> UNIQUE Constraint: A Column declared with UNIQUE constraint want allow repetition of values, however, NULL Values will be allowed.

3> PRIMARY KEY Constraint: A PRIMARY KEY Constraint is a combination of NOT NULL & UNIQUE. i.e. A Column declared with this constraint can neither contain NULL nor DUPLICATE values. Each table may contain only one PRIMARY KEY.

4> REFRENTIAL INTIGRITI Constraint (Foreign Key): A Foreign Key is child column to other parent column. The Parent Column has to be Either the PRIMARY KEY Columns or UNIQUE KEY Column. The Parent columns may be of a diffrent table or same table.

5> CHECK Constraint: A Check Constraint is any user-defined conditions.

Defining Constraints

```
SQL> Create table <table_name>
        (column1 datatype CONSTRAINT <constraint_name> CONSTRAINT_TYPE,
         Column2 datatype CONSTRAINT <constraint_name> CONSTRAINT_TYPE,
         Column3 datatype CONSTRAINT <constraint_name> CONSTRAINT_TYPE
        );
SQL> create table bank
            (bid number(4) CONSTRAINT bid_pk PRIMARY KEY,
            city varchar2(20) CONSTRAINT city_nn NOT NULL,
            state varchar2(20) CONSTRAINT state_nn NOT NULL
            );
SQL> create table customer
            (accno number(6) CONSTRAINT acc_pk PRIMARY KEY,
             Name varchar2(20) CONSTRAINT name_nn NOT NULL,
            Address varchar2(20) CONSTRAINT add_nn NOT NULL,
            Pan_no number(10) CONSTRAINT pan_u UNIQUE,
            Bal number(12) CONSTRAINT bal_c CHECK(bal > 500),
            Bid number(4) CONSTRAINT bid_FK
                            REFERENCES bank(bid)
            );
```

Options with Foreign Key:

```
SQL> create table customer
            (accno number(6) CONSTRAINT acc_pk PRIMARY KEY,
             Name varchar2(20) CONSTRAINT name_nn NOT NULL,
            Address varchar2(20) CONSTRAINT add_nn NOT NULL,
            Pan_no number(10) CONSTRAINT pan_u UNIQUE,
            Bal number(12) CONSTRAINT bal_c CHECK(bal > 500),
            Bid number(4) CONSTRAINT bid_FK
                            REFERENCES bank(bid)
                            ON DELETE CASCADE | ON DELETE SET NULL
            );
```

7.2 Adding a Constraint by Using Alter Command

Alter table table_name add[constraint constraint_name] constraint_type (column).

Note: 1. We can add or drop a constraint, but we cannot modifying a constraint.
2. We can enable or disable constraint.
3. We add a not null constraint by using the modify clause.

Add a foreign key constraint to the emp table indicating that a manager must already exist as a valid employee in the emp table.

```
SQL> alter table cust
add constraint bid_fk foreign key (bid) references bank (bid);

SQL> alter table cust
        Drop constraint name_nn;

SQL> alter table cust
        Modify (name varchar2(20) CONSTRAINT name_nn NOT NULL);
```

Viewing Constraints

Query the USER_CONSTRAINTS table to view all constraint definition and names.
Query the USER_CONS_COLUMNS view columns associated with the constraint names.

Questions

1. Why does the primary key constraint automatically enforce a NOT NULL constraint.
2. Write the syntax for inserting the selective column in a table.
3. Write the command to change the department of king to 40.
4. Write the command to give everybody a commission of 500.
5. Delete the records of clerk.
6. Write the statement that you can use to drop the unused columns from table emp.
7. Revoke UPDATE privilege from every body on the emp table.
8. Write a statement to grant INSERT privilege on the empno column of the emp Table to SCOTT.
9. Modify the sal column of the emp table to not null and increase its size to 10.
10. Display columns and constraints of table emp.
11. Write a command to delete the table.
12. Modify the emp table add constraint CHECK for SAL of TABLE EMP (condition will be the salary should greater than 2500).
13. Add a column to the existing table emp, which will hold the grades for each employee.
14. While working, set appropriate savepoints. Do a couple COMMIT and ROLLBACK operations and record you observations.
15. What is referential integrity?

8. DATABASE OBJECTS

The various database objects are:

- Synonym
- Sequence
- Views
- Indexes

8.1 Synonyms

A synonyms is a database object, which is used as an alias (alternative name) for a table, views or sequence. Synonym can be private or public. The former is created by normal user, which is available only to that person whereas the latter is created by DBA, which can be availed by any database user.

A synonym is an alias for any table, views, materialized view, sequence, procedure, function, or pacakage. Because a synonym is simply an alias, it requires no storage other than its definition in the data dictionary.

Synonyms are often used for security and convenience. For example, they can do the following:

- Mask the name and owner of an object
- Provide location transparency for remote objects of a distribute database
- Simplify SQL statements for database users

You can create both public and private synonyms. A public synonym is owned by the special user group name PUBLIC and every user in a database can access it. A private synonym is in the schema of a specific user who has control over its availability to others.

Synonyms are very useful in both distributed and nondistributed database environment because they hide the identity of the underlying object, including its location in a distributed system. This is advantageous because if the underlying object must renamed or moved, then only the synonym needs to be redefined. Application based on the synonyms continue to function without modification.

Syntax:

SQL> Create synonym e for emp;

When a grant is given to another user on this synonym he can perform DML operation such as insert, delete, update, on the synonym in the same way that of the table but he cannot perform any DDL operations on the synonym except dropping the synonym.

The synonym is just an alias of the table and all the manipulations on it actually effect the table.

Public synonyms are created by database administrator to hide the identity of the base table and reduce the complexity of SQL statements. One such public synonym is TAB, which we use for selecting the tables owned by the user. These public synonyms are owned by user group PUBLIC.

Note:
To get the details of the sequence that the user has created we use the data dictionary view USER_SYNONYMS.

8.2 Sequences

A sequence is a database object, which can generate unique, sequential integer values. It can be used to automatically generate primary key or unique key values. A sequence can be either in an ascending or a descending order.

A sequence generate a serial list of unique numbers for numeric columns of a database's tables. Sequences simplify application programming by automatically generating unique numerical values for the rows of a single table multiple tales.

For example, assume two users are simultaneously inserting new employee rows into the EMP table. By using a sequence to generate unique employee numbers for the EMPNO column, neither user has to wait for the other to enter the next available employee number. The sequence automatically generates the correct values for each user.

Sequence numbers are independent of tables, so the same sequence can be used for one or more tables. After creation, a sequence can be accessed by various users to generate sequence numbers.

Syntax:

```
Create sequence <seq_name>
[increment by n]
[ start with n]
[ maxvalue n]
[minvalue n]
[cycle/noncycle]
[cache/nocache]
```

While creating a sequence we can define the following, start with increament by n, n is an integer which specifies the interval between sequence numbers.

The default is 1.

Start with	specifies the first sequence numbers to be generated.
Minvalue	Specifies the minimum value of the sequence.
Maxvalue	Specifies the maximum value that the sequence can generate.
Cycle	Specifies that the sequence continues to generate values from the beginning after reaching either its max or min value.
No cycle	specifies that the sequence cannot generate morevalues after reaching either its maxvalue or minvalue. The default is 'no cycle'.

Cache	The CACHE option pre-allocates a set of sequence numbers and retains them in memory so that sequence numbers can be assigned faster. When the last of the sequence numbers in the cache has been used, Oracle reads other set of number into the cache.
Nocache	The default value 'noncache', does not preallocate sequence numbers for faster access.

E.g.:

```
SQL>create sequence stdid
Increment by 1
Start with 1
Maxval 10
Minval 1
Cycle
Cache 4;
```

A pseudocolumn behaves like a table column, but is not actually stored in the table. You can select from pseudocolumn, but you cannot insert, update, or delete their values.

After creating the sequence we can access it with the pseudo columns like currval and Nextval. Nextval returns initial value of the sequence, when referred to, for the first time.

Currval returns the current value of the sequence which is the value returned by the last reference to nextval.

```
SQL> select stdid.nextval 30;
```

Note: A sequence can be altered to change the increment value or to change the number of cached sequence numbers.

```
SQL> alter sequence stdid maxval 30;
```

Note: To get the details on sequences that user has created we query the data dictionary view USER SEQUENCES.

CURRVAL	The CURRVAL pseudocolumn returns the current value of a sequence.
NEXTVAL	The NEXTVAL pseudocolumn increments the sequence and returns the next values.

You must qualify CURRVAL and NEXTVAL with the name of the sequence:

Sequence. CURRVAL
Sequence. NEXTVAL

To refer to the current or next value of a sequence in the schema of another user, you must have been granted either SELECT object privilege on the sequence or SELECT ANY SEQUENCE system privilege, and you must qualify the sequence with the schema containing it.

Schema.sequence.CURRVAL
Schema.sequence.NEXTVAL

Where to Use Sequence Values

You can use CURRVAL and NEXTVAL in:

- The SELECT list of a SELECT statement that is not contained in a subquery, materialized view, or view
- The SELECT list of a subquery in an INSERT statement
- The VALUES clause of an INSERT statement
- The SET clause of an UPDATE statement

Restrictions: You cannot use CURRVAL and NEXTVAL :

- A sub query in a DELETE, SELECT, or UPDATE
- A query of a view or of a materialized view
- A SELECT statement with the DISTINCT operator
- A SELECT statement with a GROUP BY clause or ORDER BY clause
- A SELECT statement that is combined with another SELECT statement with the UNION, INTERSECT, or MINUS set operator
- The WHERE clause of a SELECT statement
  DEFAULT value of a column in a CREATE TABLE or ALTER TABLE statement
- Condition of a CHECK constraint

Also, within a single SQL statement that uses CURRVAL or NEXTVAL, all referenced LONG column, updated tables, and locked tables must be located on the same database.

How to use sequence values
When you create a sequence, you can define its initial value and the increment between its values. The first reference to NEXTVAL returns the sequence's initial value. Subsequent references to NEXTVAL increment the sequence value by the defined increment and return the new value. Any reference to CURRVAL always returns the sequence's current value, which is the value returned by the last reference to NEXTVAL. Note that before you use CURRVAL for a sequence in your session, you must first initialize the sequence with NEXTVAL.

Within a single SQL statement, Oracle will increment the sequence only once for each row. If a statement contains more than one reference to NEXTVAL for a sequence, Oracle increment the sequence once and returns the same value for all occurrence of NEXTVAL. If a statement contain references to both CURRVAL and NEXTVAL, Oracle increments the sequence and returns the same value for both CURRVAL and NEXTVAL regardless of their order within the statement.

A sequence can be accessed by many users concurrently within no waiting or locking.

8.3 VIEW

A view is a "Stored query' or "Virtual Table". We can use views in most place where a table can be used. The table upon which a view is based are called base tables.

Views provide an additional level of table security by restriction the access to a predetermined set of rows and for column of a table.

They hide data complexity i.e. a single view can be defined with a join, which is a collection of related column or rows in multiple tables. The view hides the fact that this information actually originates from several tables. They simplify commands for the user because they allow them to select information from multiple tables without actually knowing how perform a join. They isolate applications from changes in definition of base tables.

Syntax:

Create [or replace] [no] [force]] view <view_name>
as <query> [with[check option] [read only] [constraint]];

Or Replace
Specify OR REPLACE to re-create the view if it already exists. You can use this clause to change the definition of an existing view without dropping, re-creating, and regranting object privileges previously granted on it.

With Read Only
Specify WITH READ ONLY if you want to deletes, inserts, or updates to be performed through the view.

With Check option
Specify WITH CHECK OPTION to guarantee that insert and updates performed through the view will result in rows that the view subquery can select. The CHECK OPTION cannot make this guarantee if:

- There is a subquery within the subquery of this view or any view on which this view is based or
- INSERT, UPDATE, or DELETE Operations are performed using INSTEAD OF triggers

Force
Specify FORCE if you want to create the view regardless of whether the view's base tables or the referenced object types exists or the owner of the schema containing the view has privileges on them. These conditions must be true before any SELECT, INSERT, UPDATE, or DELETE statements can be issued against the view.

If the view definition contains any constraints, CREATE VIEW…. FORCE will fail if the base table does not exists or the referenced object type does not exist. CREATE VIEW….FORCE will also fail if the view definition references a constraint that does not exist.

No Force
Specify NOFORCE if you want to create the view only if the base tables exist and the owner of the schema containing the view has privileges on them. This is the default.

SQL> Create view temp as select * from emp;

The above command will create a view with name vemp, which will have the same structure of emp and all the rows of the base table are accessible through this view.

```
SQL> Create or replace view vemp
        as
        select * from emp
        where deptno=10
        with check option;
```

The following command creates a view with specific columns and we try to update any other columns it gives error.
We can even insert rows into the view, provide other column in the base table accept null values. If any of the table columns, which are not selected in view query were assigned as not null, then we cannot insert tows.

Note: USER_VIEWS will provide details on VIEWS that the user has created.

8.4 "TOP-N" Analysis

Top-n queries for the n largest or smallest values of a column

- What are the ten best selling products?
- What are the ten worst selling products?

Both the largest values and smallest values sets are considered Top_ N queries.
Performing "Top_N" Analysis.

Syntax:
Select [column_list], rownum From (select [column_list] from table order by top_n_column) Where rownum<=N;

Example:
To display top 3 earners of a company

```
SQL>select rownum as rank, ename, sal
        From (select ename, sal from emp
        Order by sal desc)
        Where rownum<= 3;
```

To display the four most senior employees of a company

```
SQL> select  rownum as senior, e.ename, e.hiredate
             From (select ename, hiredate from emp order by hiredate) emp
             Where rownum<=4;
```

8.5 INDEX

Index is optional structures associated with tables. We can create indexes explicitly to speed up SQL statement execution on a table. Similar to the indexes in books, that help us to locate information faster, an Oracle index provides a faster access path to table data.

The index points directly to the location of the rows containing the value. Indexes are the primary means of reducing disk I/O when properly used appropriately. We create an index on a column or combination of columns.

Indexes are optional structures associates with table and clusters. You can create indexes on one or more columns of a table to speed SQL statement execution on that table. Just as the index in this manual helps you location information faster than if there were no index, an Oracle index provides a faster access path to table data. Index are the primary means of reducing disk I/O when properly used.

You can create many indexes for a table as long as the combination of columns differs for each index. You can create more than one index using the same columns if you specify distinctly different combination of the column.

For Example: the following statements specify valid combination:

```
SQL>create index emp_indx1 on emp (ename, job);
SQL>create index emp_indx2 on emp (job, ename);
```

You cannot create an index that references only one column in a table if another such index already exists. Oracle provides several indexing schemes, which provide complementary performance functionality:

- B-tree indexes
- B-tree cluster indexes
- Hash cluster indexes
- Reverse key indexes
- Bitmap indexes
- Bitmap join indexes

Oracle also provides support for function-based indexes and domain indexes specific to an application or cartridge.

The absence or presence of an index does not require a change in the wording of any SQL statement, An index is merely a fast access path to the data. It affects only the speed of execution. Given a data value that has been indexes. The index points directly to the location of the rows containing that value.

Indexes are logically and physically independent of the data in the associated table. You can create or drop an index at any time without affecting the base tables or other indexes. If you drop an index, all application continue to work. However, access of previously indexed data can be slower. Indexes, as independent structures, requires storage space.

Oracle automatically maintains and uses indexes after they are created. Oracle automatically reflects changes to data, such as adding new rows, updating rows, or deleting rows, in all relevant indexes with no additional action by users.

Retrieval performance of indexed data remains almost constant, even as new rows are inserted. However, the presence of many indexes on a table decreases the performance of updates, deletes, and insert, because Oracle must also update the indexes associate with the table. The optimizer can use an existing index to build another index. This result in a much faster index build.

Unique and Nonunique indexes
Indexes can be unique or nonunique. Unique indexes guarantee that no two rows of a table have duplicate values in the key column (or columns). Nonunique indexes do not impose this restriction on the column values.

Oracle recommends that unique indexes be created explicitly, and not through enabling a unique constraint on a table.

Alternatively, you can define UNIQUE integrity constraints on the desired columns. Oracle enforce UNIQUE integrity constraints by automatically defining a unique index on the unique key. However, it is advisable that any index that exist for query performance, including unique indexes, be created explicitly.

8.6 Data Dictionary

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. For example, a data dictionary stores information about both the logical and physical structures of the database. In addition to this valuable information, a data dictionary also stores such information as:

- The valid users of an Oracle database.
- Information about integrity constraints defined for tables in the database.
- How much space is allocated for a schema object and how much of it in use.

A data dictionary is created when a database is created. To accurately reflect the status of the database at all times, the data dictionary is automatically updated by Oracle in response to specific action (such as when the structure of the database is altered). The data dictionary is critical to the operation of the database, which relies on the data dictionary to record, verity, and conduct ongoing work. For example, during database operation, Oracle reads the data dictionary to verify that schema objects exists and that users have proper access to them.

Note: 1. To list all data dictionary views accessible to the user.

```
SQL>select * from dictionary;
```

2.To view a description on of each column in data dictionary tables and views, query the DICT_COLUMNS view.

```
SQL>select column_name comments from dict_columns where
        table_name='USER_OBJECT';
```

8.7 ROWID

For each row in the database, the ROWID pseudocolumn return a row's address. Oracle9i rowid values contain information necessary to locate a row:
- The data object number of the object
- Which data block in the datafile
- Which row in the data block (first row is 0)
- Which datafile (first file is 1). The file number is relative to the tablespace.

Usually, a rowid value uniquely identifies a row in the database. However, rows in different tables that are stored together in the same cluster can have the same rowid.

Values of the ROWID pseudocolumn have the datatype ROWID rowid values have several important uses:

- They are the fastest way to access a single row.
- They can show you how a table's rows are stored.
- They are unique identifiers for rows in a table.

You should not use ROWID as a table's primary key. If you delete and reinsert a row with the import and export utilities, for example, its rowid may change. If you delete a row, Oracle may reassign its rowid to a row inserted later.

Although you can use the ROWID pseudocolumn in the SELECT and WHERE clause of query, these pseudocolumn values are not actually stored in the database. you cannot insert, update, or delete value of the ROWID pseudocolumn.

This statement selects the address of all rows that contain data for employees in department 20;

```
SQL> select rowid, ename from emp where deptno = 20;
```

8.8 ROWNUM

For each row returned by a query, the ROWNUM pseudocolumn returns a number indicating the order in which Oracle selects the row from a table or set of joined rows. The first row selected has a ROWNUM of 1, the second has 2, and so on.

You can use ROWNUM to limit the number of rows returned by a query, as in this example:

```
SQL>select *from emp where rownum<10;
```

If an ORDER BY clause follows ROWNUM in the same query, the rows will be reordered by the ORDER BY clause. The results can vary depending on the way the rows are accessed. For example, if the ORDER BY clause causes Oracle to use in index to access the data, Oracle may retrieve the rows in a different order than without the index. Therefore, the following statement will not have the same effect as the preceding example:

```
SQL>select *from emp where rownum <4 order by sal;
```

If you embed the ORDER BY clause in a subquery an place the ROWNUM condition in the top-level query, you can force the ROWNUM condition to be applied after the ordering of the rows. For example, the following query returns the top 3 earners in a company. This is sometimes referred to as a "top-N query";

```
SQL> select * from(select *from emp order bysal)
        Where rownum <4;
```

In the preceding example, the ROWNUM values are those of the top-level SELECT STATEMENT, so they are generated after the rows have already been ordered by sal in the subquery.

Questions

1. How does a view differ from a table.
2. Create a view empview, which will contain the empno, ename, sal, deptno and dname.
3. In what situation must a sequence be used.
4. Create index on ename column of emp table.
5. Delete the index that are no longer needed.
6. What is the data dictionary view to see the sequence.
7. What is a synonym? How is it used?

9. PRACTICE QUESTIONS

DEPT TABLE :

Dept No.	Dname	LOC
20	Research	Dallas
10	Accounting	New York
30	Sales	Chicago
40	Operations	Boston

EMP TABLE :

Empno	Ename	Job	MGR	Hiredate	Sal	COMM	Dept No.
7369	Smith	Clerk	7902	17-Dec-80	800		20
7499	Allen	Salesman	7698	20-feb-81	1600	300	30
7521	Ward	Salesman	7698	22-Feb-81	1250	500	30
7566	Jones	Manager	7839	O2-Apr-81	2975		2-
7654	Martin	Salesman	7698	28-Sep-81	1250	1400	30
7698	Blake	Manager	7893	01-May-81	2850		30
7782	Clark	Manager	7839	09-Jun-81	2450		10
7788	Scott	Analyst	7566	09-Dec-82	3000		20
7839	King	President		17-Nov-81	5000		10
7844	Turner	Salesman	7698	08-Sep-81	1500	0	30
7876	Adams	Clerk	7788	12-Jan-83	1100		20
7900	James	Clerk	7698	03-Dec-81	950		30
7902	Ford	Analyst	7599	03-Dec-81	3000		20
7934	Miller	Clerk	7782	23-Jan-82	1300		10

SAL GRADE TABLE :

Grade	Losal	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

SELECT :

1. Display all the information of the EMP table.
2. Display unique Jobs from EMP table.
3. Display the details of all MANAGERS.
4. List the emps who joined before 1981.
5. Display the Empno, Ename, Job, Hiredate, and experience of all Managers.
6. List the Empno, Ename, Sal, Exp of all emps working for Mgr 7639.
7. Display all the details of the emps whose Comm. Is more than their Sal.
8. List the emps along with their Exp and whose Daily Sal is more than Rs. 100.
9. List the emps who are working for the Deptno 10 or 20.
10. List the emps who are working under any Manager.
11. List all the Clerks of Deptno 20.
12. Display the details of SMITH.
13. Display the Empno, Ename, Deptno from EMP table.
14. Write a query to display the Empno and the Deptno of all emps.
15. Display the unique Depts of emps.
16. List the emps whose Salary is more than 3000 after giving 20% increment.
17. List the Ename and Sal increased by 15% and expressed as No. of Dollars.
18. Produce the output of EMP table EMP_AND_JOB for Ename and job.
19. Display the Empno, Ename, Salary of all Managers.
20. Define a variable representing the expression used to calculate on emps total annual remuneration use the variable in a statement which finds all emps who can earn 30000 a year or more.
21. Check whether all the emps numbers are indeed unique.
22. List the Empno, Sal and Comm of emps.
23. Display the unique dept with Jobs.
24. Display the details of Blake.
25. List all Clerks.
26. List all emps joined on 1 may 1981.
27. List the emps whose Salaries are less than 3500.
28. List the emps Empno, Ename, Sal of all emp joined before 1 Apr 1981.
29. List the emps whose exp is more than 10 years.
30. List the emps who are working as Manager.
31. List the emps who are working as clerks and exp is more that 8 Years.
32. List the Empno, Ename, Sal, Job of emps with the annSal <34000 but receiving some comm. Which should not be greater than Sal and the designation should be Salesman working for dept 20.
33. List all the Salesmen who are receiving Comm.
34. List all the Salesmen of the Dept 30 whose Comm. Is more than their Salary.
35. Generate all the 3 digit numbers.
36. Display the first 5 records of the EMP table.
37. List the emps empno, ename, job, sal of all emps.
38. List all the unique deptno of emps.
39. List all the unique jobs along with deptno.
40. List all the details of ‘Miller’.
41. list the details of dept 10.
42. List all the ‘SALESMAN’.
43. List all the emps who joined before 1984.
44. List all the emps whose Sal>2500.

45. List all the emps who are working since 1st April 1982.
46. List the empno, ename, sal, Daily sal of all emps.
47. List the empno, ename, sal, experience of all ‘Analysts’.
48. List the emps whose exp>6.5 Y.
49. List the emps who joined in 2nd half of 1981.
50. List all the emps in dept 10 who are working as Clerks.
51. List the emps who joined before 1985 and salary is more than 3000.
52. List the Exp of Grade 3 emps.

SUB QUIRES:

53. List the details of the emps whose salaries more than the employee BLAKE
54. List the emps whose Jobs are as ALLLEN.
55. List the emps who are senior to King.

UPDATE:

56. Transfer Blake to Deptno 30.
57. Transfer the emps of Dept 10 to 20.
58. Transfer the emps Chicago to Dallas.
59. Update the Salary of ‘ALLEN’ with the highest paid emp of Grade 2 and transfer him to Blake’s Dept and change the Mgr to Blake.
60. Increment the Salaries by 2% and add a Comm of 250 to the existing Comm and changing the Manager to Jobs of all ‘Salesman’ whose Salaries os more than or equal to 1000.
61. Replace the Sal of most senior ‘CLERK’ with the most senior employee Sal of Grade 3.
62. Transfer all the emps of 10-20-,20-30,30-40,40-10.
    a. Update the Salary of each employee by 10% increment who are not eligible for Comm.
63. Increment the Sals of the emps be 2%.
64. Increment the Sal of all the Clerks by Rs.200 and change their Mgr to 7654.
65. Change the Mgr of all Chicago related emps Blake.
66. Write update statement to increment the Sal of grade 2 by 1.2%.
67. Transfer the emps to analyst Dept and give the Sal of smith plus 500 to those belongs to grade 3 working at New York or Dallas with an exp>7y whose name should not be 4 chars.
68. Increment the salaries of all Clerks by 2%.
69. Transfer the emps Dept 10 to 20.
70. Update Mgr 7788 to those are working under the Mgr 7839 & increase the sal by 2%.
71. Transfer the emps of Chicago to Dept 20.
72. Update the Deptno of Sales dept for those emps working at Deptno 10.
73. Update the salary of Smith with the highest paid salary emp of Salesman more than 10Y exp.
74. Increment the sal of grade 2 emps by Rs.300.
75. Update the sal of Allen with the highest Sal of any Clery belongs to grade 2 or 3 working at Chicago or Dallas and king as Manger.
76. Change the mgr to 7788 for those working for the mgr 7369.
77. Add 250 to the comm of all salesmen who are receiving some comm..

78. Change the deptno of Blake to 30 and also change the job as President with an increment in the Salary%.
79. Transfer the emps of dept no 20 to Sales dept.
80. Replace the Sal of Smith with the highest paid Salesman with exp>10 years.
81. Give an increment of Rs.300 to all emps of grade 2.
82. Replace the Sal of Allen with the highest paid Clerk of New York or Chicago.

DELETE:

83. Delete all the information of ‘Salesmen’.
84. Delete all the Managers who are working under king the Sal ranging from 3500 to 4000 joined after second half of 981.
85. Delete all Grade 1 emps.
86. Write a query to delete all the Salesman of Sales dept.
87. Delete the most recently hired employee of Deptno 30.
88. Delete the records where no of emps in a particular dept is less than 3.
89. Delete the emps who joined in the company before 10 years back.
90. Delete all the emps working in Boston.
91. Delete the emps who joined in the month of Jan 1981.
92. Delete all the emps who working under Blake with exp>7year.
93. Delete the information of Miller.
94. Delete all emps working under Blake.
95. Delete the emps who joined most recently under king.
96. Delete the Grade 1 and 2 emps.
97. Delete the emps who belongs to Grade 1 or 2 working at Chicago joined in any month of the first half of 81.
98. Delete the emps of Mrg 7654.
99. Delete the emps with experience <4 and whose Job is ending with ‘MAN’.
100. Delete the emps of Sales Dept whose Salary ranging from 1500 to 3000.
101. Delete all grade 2 & 3 emps of Sales Dept and working at ‘Boston’ joined after 1983.
102. Delete the information of James.
103. Delete the emps in dept 10 or 20.
104. Delete the transformation of emps who are working at Chicago with exp<7Y & whose name is not having 4 chars.
105. Delete the emps whose salaries are lowest sals of their own dept.
106. Delete the emps who are senior to their own Mgrs.
107. Delete the emps who are at DALLAS and they are in dept 30.
108. Delete those dept’s where no employee is working.
109. Delete duplicate records in the emp table.
110. Delete the info. Of emps whose exp <4 years.
111. Delete all Salesmen with comm.<250 and also those are not receiving comm..
112. Delete the emps belongs to grade 1 except the Clerks.

JOINS :

113. List the total information of EMP table along with Dname and Loc of all the emps working under ‘ACCOUNTING’ or ‘RESEARCH’ in the asc Deptno.

114. List the Empno, Ename, Sal, Dname of al the ‘MANAGER’ and ‘ANALYST’ working in New York or Dallas with an exp more than 7 years without receiving the Comma sc order of Loc.
115. Display the Empno, Ename, Sal, Dname, Log, Deptno, Job of all emps working at CHICAGO or working for ACCOUNTING dept with Ann sal >28000, but the Sal should not be = 3000 or 2800, Who doesn’t belongs to the Manager and Whose No. having a digit ‘7’ or ‘8’ in 3rd position in the asc order of Deptno and desc order of Job.
116. Display the total information of the emps along with Grades I the asc order of grade.
117. List all the grade 2 and Grade 3 emps.
118. Display all grade 4.5 Analyst and Manager.
119. List the Empno, Sal, Dname, grade, Experience and Annual sal of emps working for Dept 10 or 20.
120. List the details of the Depts along with Empno, Ename or without the emps.
121. List the emps whose are senior to their own Manager.
122. List the emps of Deptno 20 whose Jobs are same as Deptno 10.
123. List the emps whose Salary is same as FORD or SMITH in desc order of Salary.
124. List the emps whose Jobs are sale as MILLER or Sal is more than ALLEN.
125. List the emps whose Sal is more than the total remuderation of the SALESMAN.
126. List the emps who are senior to BLAKE working at CHICAGO & BOSTON.
127. List the emps of Grade 3,4 belongs to the dept ACCOUNTING and RESEARCH whose Sal is more than ALLEN and exp more than SMITH in the asc order of Exp.
128. List the emps whose jobs same as SMITH or ALLEN.
129. List the most recently hired employee of Grade 3 belongs to the Loc CHICAGO.
130. List the emps with their Dept names.
131. List the emps who are not working in ‘Sales’ dept.
132. List the emps Name, Dept, Sal & Comm for those whose Salary is between 2000 and 5000 and Loc is Chicago.
133. List the emps whose Salary is greater than his Manager’s salary.
134. List the Grade, Ename for the Deptno 10 or 30 but grade is not 4 while they joined in the company before ’31-Dec-82’.
135. List the ename, Job, Dname, Loc for those who are working as a MANAGERS.
136. List the emps whose Manager name is ‘Jones’ & also List their Manager names.
137. List the Name and Salary of Ford if his Salary is equal to Hisal of his Grade.
138. List the Name, Job, Dname, Mgrno, Salary, grade Dept wise.
139. List the emps Name, Job and MGR without Manager.
140. List the Name, Salary, Comm for those emps whose Net pay is greater than or equal to any other employee Salary of the company.
141. List the emps whose Sal is less than his Manager but more than other Managers.
142. List the Mgrs who are not working under ‘President’ but working under other Manager.
143. List the emps who joined in the company on the same date.
144. List the employee Name, Job, Annual salary, Deptno, dept name, and grade who earn 36000 a year and who are not CLERKS.
145. List the Name, Job and Salary of the emps who are not belonging to the department 10 but who have the same Job or Salary as the emps of dept 10
146. List the details of emps working at Chicago.
147. List the Empno, Ename, deptno, loc of all emps.

148. List the Empno, Ename, loc of dname of the depts.10 or 20.
149. List the empno, Sal, loc of emps working at Chicago sal as with an exp>6y.
150. List the emps along with of those who belongs to Dallas New York with Sal ranging from 2000 to 5000 joined in 1981.
151. List the Empno, Ename, Sal, grade of all emps.
152. List the grade 2 & 3 emps of Chicago.
153. List the emps with loc & grade of Accounting dept or the location Dallas or Chicago with the grades 3 to 5 & exp >6y.
154. List the grades 3 emps of Research or Operations depts. Joined after 1987 and whose names should not be either Miller or Allen.
155. List the emps whose Job is same as Smith.
156. List the emps who are senior to Miller.
157. List the emps whose Job is same as either Allen or Sal is more than Allen sal.
158. List the emps whose Sal is greater than Blake salary.
159. List the dept 10 emps whose Sal is more than Allen Salary.
160. List the Managers who are senior to king & who are junior to Smith.
161. List the Empno, Ename, loc, Sal, Dname of all emps belongs to king dept.
162. List the emps whose grade is greater than the grade of Miller.
163. List the emps who are belongs Dallas or Chicago with the grade same as Adams or exp more than Smith.
164. List the emps whose Sal is same as Ford or Blake.
165. List the emps details of the emp of Chicago.
166. List the Empno, Ename, Sal, Experience, TA, HRA, DA and Gross and Dname of all emps joined in the year 1981 in the Loc NEWYORK and DALLAS with Sal <4500.
167. List Empno, Ename, Sal, Dname, Loc of all emps working at DALLAS, with the Sal should not more than 3500 in the desc order of sal.
168. List the information of EMP along with Loc, employee working at Loc which does not have a char 'a' with the annual Salary ranging from 22000 to 40000 without Comm.
169. List the Empno, Ename, Deptno and Dname of all emps belongs to RESEARCH Dept.
170. List all the emps along with Location who belongs to Depts 10 or 20.
171. List the emps whose Sal is more than Blake Sal.
172. List the emps of Deptno 10 whose Jobs are same as any emp job of deptno 20.
173. List the emps who are senior to Ward.
174. List the empno, ename, Sal, Grade of all emps who belongs to the grade 3 or 4
175. List the Empno, Ename, Sal, Grade of all emps working for the mgrs 7788, 7369, 7566.
176. List all the Grade 3 'Analysis and 'managers' in the asc order of Job.
177. List all Grade 4 emps working for Research Dept along with Grade and Dname.
178. List the emps working at Chicago with the job same as the emp of DALLAS.
179. List the emps of NEWYORK whose Job is same as any emp belongs to grade 3.
180. List the details of the emps whose Jobs are same as FORD and BLAKE.
181. List the empno, ename, sal, deptno, all emps working in Accounting or sales dept.
182. List the unique job categories of emps working at Dallas in the dest order.
183. List the emps along with the dname, those are working in location which is not having a char 'Y' or blank space in the ename & exp>15 Y & does not belongs the jobs president Salesman in the asc order of exp.

184. List the empno, ename, annsal, job, loc of all emps working at Chicago, Dallas with an annsal between 16000 to 52000 & whose job != Analyst working with some Mgr joined in jan to June & Nov-Dec but not belongs to year 83 or 84 in the asc order of loc.
185. List the empno, ename, sal, total details of the dept where the emps sal>3/4th of comm. And working in dept which is not located in Boston, Dallas & whose names are not having a chr 'Z' or 'TH' or 'LL' together & sal is 4 digit & it should not ending with 0 in the asc order of loc.
186. List all Managers & Analyst of grade 3-5 with exp 8-14 Years in the asc order of grade.
187. List the details of emps with dname, grade those who are working in Accounting belongs to grade 3.
188. List the emps with exp & annsal of those belongs to grade 1,2,3 & desig as Clerk OR Salesman at Chicago or NEW YORK working for the Mgr is not ending with '9'.
189. List the emps whose sal is more than Jones sal.
190. List the details of dept where President is working.
191. List the emps along with dname whose dname is same as Allen.
192. List the emps whose grade is same as Miller grade
193. List the empno, ename, sal, job, deptno, loc, grade of emps whose grade is same as the emps who are working under Blake.
194. List the emps whose jobs are same as emps job of dept 20.
195. List the emps whose jobs are not same as any employee of sales dept.
196. List the emps whose Sal is more than any emp of dept 10.
197. List the emps who are senior to all emps & who are Mgrs to others.
198. List the emps whose jobs are same as Ford or Blake.
199. List the common jobs of grade 1 & emps working at Chicago.
200. List the jobs of emps available in Chicago & not available in Dallas.
201. List those Managers who are getting the sal less than his own emps Salary.
202. Find out the emps who joined in the company before their Managers.
203. List the Deptno where there are no emps.
204. List the emps of dept 20 & 30 whose experience is more than any employee of Dept. 10.
205. Display the emps whose Grade is 3.
206. Display the Grade of 'Jones'.
207. Display the emps who are in 'Sales' Dept not in Grade 3.
208. Display the emps who do not have any persons working under them.
209. Display the emps whose job is not 'Manager' but they are managers to some other employees.
210. Display the Dept's whose name start with's' while the Loc name ends with 'k'.
211. Print the details of all the emps who are sub-ordinates to Blake.
212. List the emps who are working as Managers using co-related sub-query.
213. List the emps whose Managers name is 'Jones' and also with their managers name.

Best of Luck

Best of Luck