The Relational Database Management System, or **RDBMS** in short, manages relational data. Oracle Database is an RDBMS with the largest market share.

d

Besides the Oracle Database, there are other RDBMS products available. Here are some notable ones:

* Db2 from IBM.
* SQL Server from Microsoft.
* MySQL – the most popular open-source database, also from Oracle.
* PostgreSQL – the most advanced open source database.
* Oracle database is a relational database management system. It is known as Oracle database, OracleDB or simply Oracle. It is produced and marketed by Oracle Corporation..
* Oracle database is the first database designed for enterprise grid computing. The enterprise grid computing provides the most flexible and cost effective way to manage information and applications.

History of oracle:

* **Larry Ellison** and his two friends and former co-workers, **Bob Miner** and **Ed Oates**, started a consultancy called **Software Development Laboratories (SDL)** in 1977.
* SDL developed the original version of the Oracle software. The name *Oracle* comes from the code-name of a **CIA-funded project**
* Ellison had worked on while previously employed by .

Oracle Database features

Oracle Database allows you to quickly and safely store and retrieve data. Here are the integration benefits of the Oracle Database:

* Oracle Database is cross-platform. It can run on various hardware across operating systems including Windows Server, Unix, and various distributions of GNU/Linux.
* Oracle Database has its networking stack that allows application from a different platform to communicate with the Oracle Database smoothly. For example, applications running on Windows can connect to the Oracle Database running on Unix.
* ACID-compliant – Oracle is ACID-compliant Database that helps maintain data integrity and reliability.
* Commitment to open technologies – Oracle is one of the first Database that supported GNU/Linux in the late 1990s before GNU/Linux become a commerce product. It has been supporting this open platform since then.

Oracle Database has several structural features that make it popular:

* Logical data structure – Oracle uses the logical data structure to store data so that you can interact with the database without knowing where the data is stored physically.
* Partitioning – is a high-performance feature that allows you to divide a large table into different pieces and store each piece across storage devices.
* Memory caching – the memory caching architecture allows you to scale up a very large database that still can perform at a high speed.
* Data Dictionary is a set of internal tables and views that support administer Oracle Database more effectively.
* Backup and recovery – ensure the integrity of the data in case of system failure. Oracle includes a powerful tool called Recovery Manager (RMAN) – allows DBA to perform cold, hot, and incremental database backups and point-in-time recoveries.
* Clustering – Oracle Real Application Clusters (RAC) – Oracle enables high availability that enables the system is up and running without interruption of services in case one or more server in a cluster fails.

Oracle Database Editions

Oracle provides three main editions of Oracle Databases as follows:

1) Enterprise Edition (EE) is the common and expensive edition of the Oracle Database. It has the following characteristics:

* No maximum number of CPUs
* No limits on memory or database size
* Include premium features that are not available in other editions.

2) Standard Edition (SE) is a limited edition of the Enterprise Edition that has the following characteristics:

* Limited to four or fewer CPUs
* No limit on memory or database size
* Include many features, but no as many as EE

3) Expression Edition (XE) is a free-to-use version of the Oracle Database that available on both Windows and GNU/Linux platforms. These are the features of Oracle Database XE 18c:

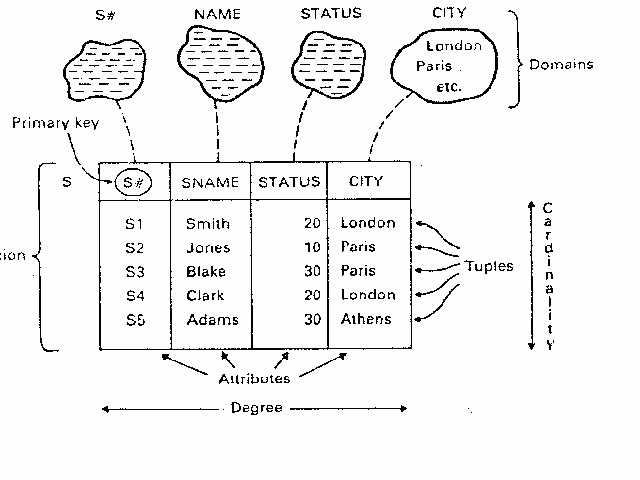
* Limited to 2 CPUs
* Can use the maximum of 2GB of RAM, and has 12GB of user data.
* Very limited features
* What is a database
* A database is an organized collection of structured data stored electronically in a computer system.
* What is RDBMS?
* RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
* A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.
* What is a table? The data in an RDBMS is stored in database objects which are called as tables. This table is basically a collection of related data entries and it consists of numerous columns and rows.
* Remember, a table is the most common and simplest form of data storage in a relational database. The following program is an example of a CUSTOMERS table:
* Below is an example of an Employee table.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Age** | **Salary** |
| 1 | Adam | 34 | 13000 |
| 2 | Alex | 28 | 15000 |
| 3 | Stuart | 20 | 18000 |
| 4 | Ross | 42 | 19020 |

### **What is a Tuple?**

* A single entry in a table is called a **Tuple** or **Record** or **Row**. A **tuple** in a table represents a set of related data. For example, the above **Employee** table has 4 tuples/records/rows.

### **What is an Attribute?**

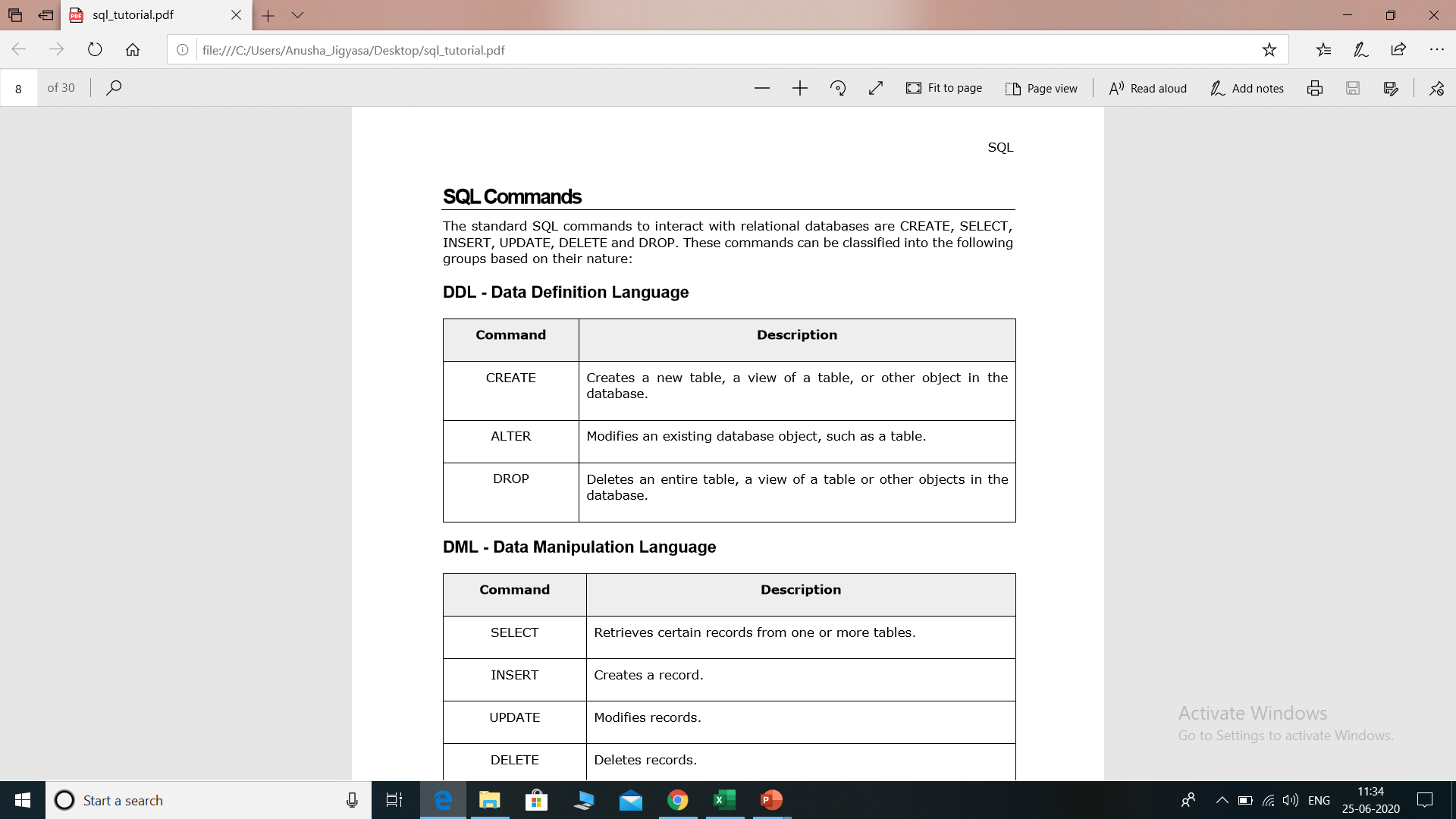
* A table consists of several records(row), each record can be broken down into several smaller parts of data known as **Attributes**. The above **Employee** table consist of four attributes, **ID**, **Name**, **Age** and **Salary**.
* 
* **Introduction to SQL,**
* SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database
* SQL is a language to operate databases; it includes database creation, deletion, fetching rows, modifying rows, etc. SQL is an ANSI (American National Standards Institute) standard language, but there are many different versions of the SQL language.
* SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.
* **Rules for SQL**
* Sql starts with a verb. E.g. Select statement
* Each verb is followed by number of clauses E.g.. From , where , Having.
* A space separates clauses E.g. Drop Table EMP.
* A semi colon(;)is used to end SQL statements.
* A statement may splits across lines but keyword may not.

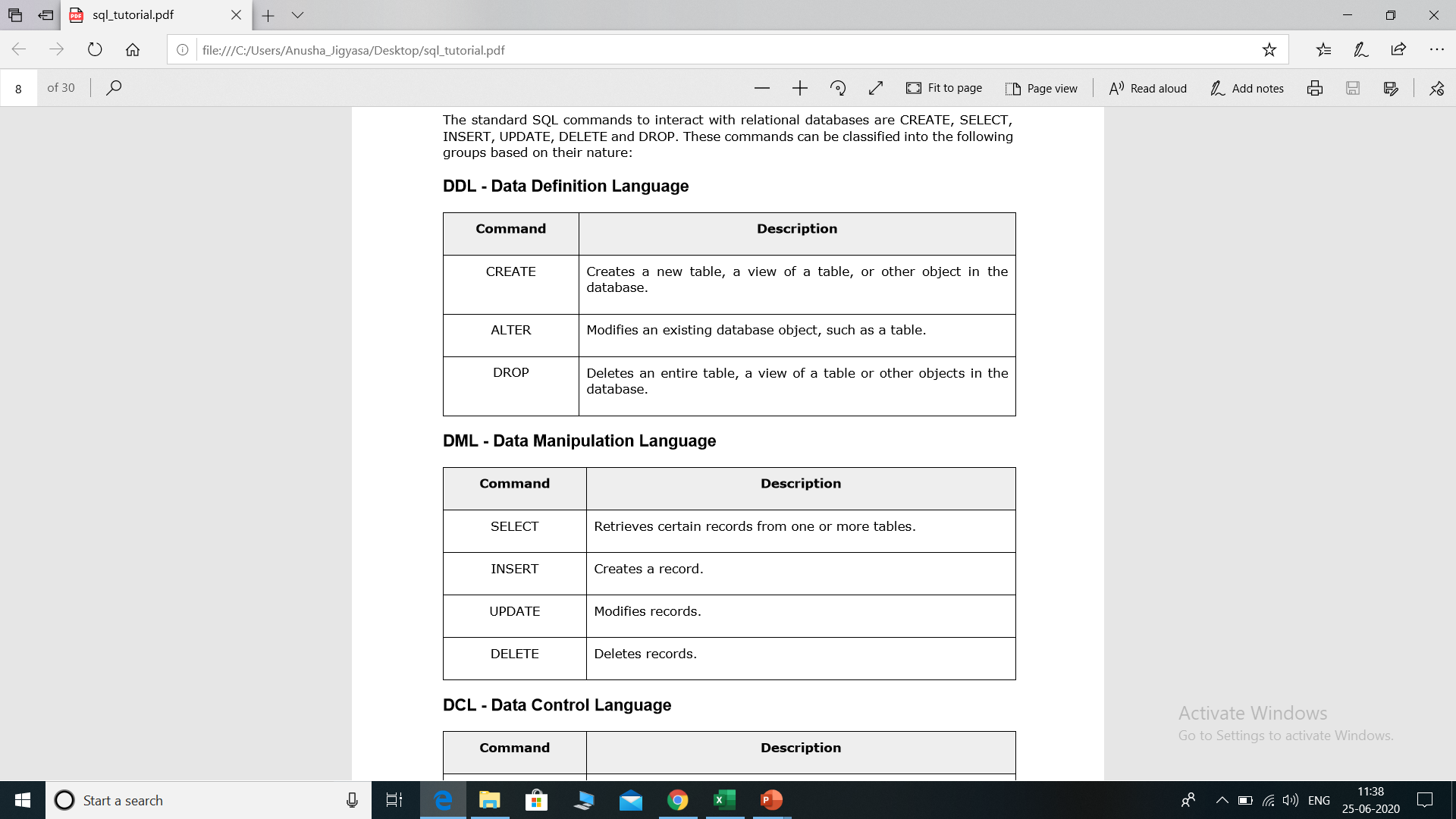
**Data Types in oracle**

1. Char(Size):-used to store character string values of fixed length. It can be hold up to 2000 char.(Oracle 10g)
2. Varchar or Varchar2:- It is used to store variable length of record alpha numeric data up to 4000 char.(oracle 10g)
3. Number(P,S):-It is used to store numbers fixed or floating. P is total size, S number of Decimal places.
4. Date :- Used to store date data. The standard date format for date is DD-MM-YY HH:MI:SS.
5. Long:- used to store variable length character up to 2GB. It can not be indexed or SUBSTR() function can not be applied on Long Data type.
6. Raw /Long raw:- Used to store Binary data Like image or Pictures
   * + - * Raw size=255 byte
         * Long raw=2GB it can not be indexed.

SQL is consists of following language

1. Data Definition Language(DDL)
2. Data Manipulation Language(DML)
3. Data Control Language(DCL)
4. Transaction Control Language(TCL)





* **DCL Commands**
  + GRANT, REVOKE.
* **TCL Commands**
  + COMMIT ,ROLLBACK, SAVEPOINT.

## CREATE TABLE statement

The CREATE TABLE is a DDL statement which is used to create tables in the database. The table gets created as soon as the CREATE TABLE script is executed and is ready to hold the data onwards The user must have the CREATE TABLE system privilege to create the table in its own schema

* (definition:-column number, data type & size).
* Syntax

CREATE TABLE <Table Name>

(Coloumn name1 Datatype(size) ,

Coloumn name2 Datatype(size),

Column nameN Datatype(size));

The Oracle INSERT statement is used to insert a single record or multiple records into a table in Oracle.

The syntax for the Oracle INSERT statement when inserting a single record using the VALUES keyword is:

INSERT INTO table

(column1, column2, ... column\_n )

VALUES

(expression1, expression2, ... expression\_n );

The most commonly used query is a SELECT query. This query is used to retrieve data from one or more tables in the database. A SELECT query is not just used alone but with it, many conditions, clauses and inner queries are used to get data from databases

SELECT query does not manipulate any data in the table on which it is executed. Select keyword in oracle is applied for fetching a set of data, which can be used singly or by combining other conditional statements as filters. When a select statement is as ‘SELECT \* from <Table\_Name>’, the whole table is displayed as the result-set, whereas select statement as ‘SELECT Column\_1, Column\_2 from <Table\_Name>’ displays the contents of only the column\_1 & column\_2 of ‘<Table\_Name>’. ‘Where’, ‘Group By’, ‘Order By’ conditions can also be applied at the end of the Select statement.

syntax

Select coloumn1,column2…….columnN from <table name>.

Example:-

**Selection of all row & all columns.**

SELECT \* FROM STUDENT

**Column wise selection**

SELECT sname, fee from student;

**Row wise selection**

#### SELECT all fields with WHERE condition

SELECT \* FROM student WHERE fee=10000;

**Row and Column wise selection**  
SELECT sname, fee from student where rollno=10;

#### **Display records in order using SELECT**

We can also display particular records in ORDER which can be ascending or descending by using the [ORDER BY clause](https://www.educba.com/order-by-in-mysql/) with the query. We will look at both ascending and descending order queries.

**a. Query for ascending order**

SELECT \* from student ORDER BY name ASC;

#### SELECT query with GROUP BY clause

We use the GROUP BY clause with SELECT statement when we want to get records based on groups. So basically it groups rows that have the same values. It is used generally in conjugation with [aggregate functions](https://www.educba.com/aggregate-functions-in-sql/). It is useful in producing summary reports.

#### SELECT query with the HAVING clause

The [having clause](https://www.educba.com/oracle-having-clause/) is used with a select statement where we want to have some conditions as where keyword cannot be used directly with aggregate functions. That is the reason the having clause was added in SQL. We are going to see an example of how we can use the having clause with a select statement.

**Query:**

SELECT COUNT(EMPLOYEE\_ID),VEHICLE\_NAME FROM employee GROUP BY VEHICLE\_NAME HAVING COUNT(EMPLOYEE\_ID)> 2;

SELECT column\_name(s)  
FROM table\_name  
WHERE ROWNUM <= number;

SELECT \* FROM Customers  
WHERE Country='Germany' AND ROWNUM <= 3;

## Integrity Constraints

Integrity constraints are used to ensure accuracy and consistency of the data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

There are many types of integrity constraints that play a role in **Referential Integrity (RI)**. These constraints include Primary Key, Foreign Key, Unique Constraints and other constraints which are mentioned below:

The following constraints are commonly used in SQL:

* [**NOT NULL**](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
* [**UNIQUE**](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
* [**PRIMARY KEY**](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [**FOREIGN KEY**](https://www.w3schools.com/sql/sql_foreignkey.asp) - Uniquely identifies a row/record in another table
* [**CHECK**](https://www.w3schools.com/sql/sql_check.asp) - Ensures that all values in a column satisfies a specific condition
* [**DEFAULT**](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column when no value is specified

## SQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

## SQL NOT NULL on CREATE TABLE

The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

### **Example**

CREATE TABLE Persons (  
    ID number(3) NOT NULL,  
    LastName varchar(25) ,  
    FirstName varchar(25) NOT NULL,  
    Age number(3)  
);

## SQL NOT NULL on ALTER TABLE

To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

ALTER TABLE Persons  
MODIFY Age number (3) NOT NULL;

## SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

## SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

CREATE TABLE Persons (  
    ID number(4) UNIQUE,  
    LastName varchar(25) NOT NULL,  
    FirstName varchar(25),  
    Age number(4)  
);

## SQL UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "ID" column when the table is already created, use the following SQL:

ALTER TABLE Persons  
ADD UNIQUE (ID);

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(25) NOT NULL,  
    FirstName varchar(25),  
    Age number (4);  
    CONSTRAINT UC\_Person UNIQUE (ID,LastName)  
);

## SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

CREATE TABLE Persons (  
    ID number (3) NOT NULL,  
    LastName varchar(25) NOT NULL,  
    FirstName varchar(25),  
    Age number (4) CHECK (Age>=18)  
);

CREATE TABLE Persons (  
    ID number (3) NOT NULL,  
    LastName varchar(25) NOT NULL,  
    FirstName varchar(25),  
    Age ID number (3) ,  
    City varchar(25),  
    CONSTRAINT CHK\_Person CHECK (Age>=18 AND City='New Delhi')  
);

ALTER TABLE Persons  
ADD CHECK (Age>=18);

ALTER TABLE Persons  
DROP CONSTRAINT CHK\_Person

## SQL DEFAULT Constraint

The DEFAULT constraint is used to provide a default value for a column.

The default value will be added to all new records IF no other value is specified.

CREATE TABLE Persons (  
    ID number (3) NOT NULL,  
    LastName varchar(25) NOT NULL,  
    FirstName varchar(25),  
    Age ID number (3)   
    City varchar(25) DEFAULT 'New Delhi'

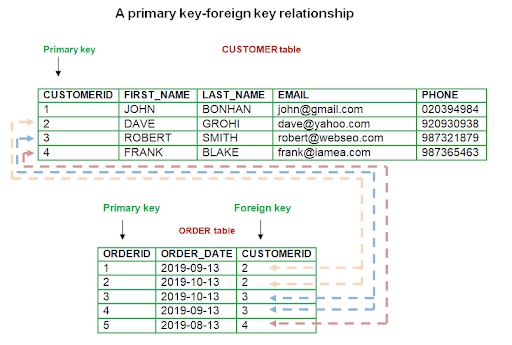
ALTER TABLE Persons  
MODIFY City DEFAULT 'Ne Delhi';

ALTER TABLE Persons  
ALTER COLUMN City DROP DEFAULT;  
);

In the relationaltonal database key is the most important element to maintain the relationship between two tables or to uniquely identify data from the table. Primary key is used to identify data uniquely therefore two rows can’t have the same primary key. It can’t be null.

On the other hand, foreign key is used to maintain relationship between two tables. Primary of a table act as forgein key in the other table. Foreign key in a table enforce Referential Integrity constraint. It can be more than one in the table.

| **Sr. No.** | **Key** | **Primary Key** | **Foreign Key** |
| --- | --- | --- | --- |
| 1 | Basic | It is used to uniquely identify data in the table | It is used to maintain relationship between tables |
| 2 | Null | It can’t be null | It can accept the null values |
| 3 | Duplicate | Two or more rows can’t have same primary key | It can carry duplicate value for a foreign key attribute |
| 4 | Index | Primary has clustered index | By default, It is not clustered index |
| 5 | Tables | Primary key constraint can be defined on temporary table | It can’t be defined on temporary tables |



## Create Primary Key - Using CREATE TABLE statement

You can create a primary key in Oracle with the CREATE TABLE statement.

Syntax

The syntax to create a primary key using the CREATE TABLE statement in Oracle/PLSQL is:

CREATE TABLE table\_name

(

column1 datatype ,

column2 datatype ,

...

CONSTRAINT constraint\_name PRIMARY KEY (column1, column2, ... column\_n)

);

Example

Let's look at an example of how to create a primary key using the CREATE TABLE statement in Oracle:

CREATE TABLE supplier

(

supplier\_id number(10),

supplier\_name varchar2(50) not null,

add varchar2(50),

CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id)

);

We could also create a primary key with more than one field as in the example below:

CREATE TABLE supplier

(

supplier\_id numeric(10) not null,

supplier\_name varchar2(50) not null,

contact\_name varchar2(50),

CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id, supplier\_name)

);

CREATE TABLE Persons (  
    ID number PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);

**Select CONSTRAINT\_NAME ,CONSTRAINT\_TYPE from user\_constraints where TABLE\_NAME='EMP'**

**Create Primary Key - Using ALTER TABLE statement**

You can create a primary key in Oracle with the ALTER TABLE statement.

Syntax

The syntax to create a primary key using the ALTER TABLE statement in Oracle/PLSQL is:

ALTER TABLE table\_name

ADD CONSTRAINT constraint\_name PRIMARY KEY (column1, column2, ... column\_n);

Example

Let's look at an example of how to create a primary key using the ALTER TABLE statement in Oracle.

ALTER TABLE supplier

ADD CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id);

In this example, we've created a primary key on the existing supplier table called supplier\_pk. It consists of the field called supplier\_id.

We could also create a primary key with more than one field as in the example below:

ALTER TABLE supplier

ADD CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id, supplier\_name);

**Drop Primary Key**

You can drop a primary key in Oracle using the ALTER TABLE statement.

Syntax

The syntax to drop a primary key using the ALTER TABLE statement in Oracle/PLSQL is:

ALTER TABLE table\_name

DROP CONSTRAINT constraint\_name;

Example

Let's look at an example of how to drop a primary key using the ALTER TABLE statement in Oracle.

ALTER TABLE supplier

DROP CONSTRAINT supplier\_pk;

In this example, we're dropping a primary key on the supplier table called supplier\_pk.

**Disable Primary Key**

You can disable a primary key in Oracle using the ALTER TABLE statement.

Syntax

The syntax to disable a primary key using the ALTER TABLE statement in Oracle/PLSQL is:

ALTER TABLE table\_name

DISABLE CONSTRAINT constraint\_name;

Example

Let's look at an example of how to disable a primary using the ALTER TABLE statement in Oracle.

ALTER TABLE supplier

DISABLE CONSTRAINT supplier\_pk;

In this example, we're disabling a primary key on the supplier table called supplier\_pk.

**Enable Primary Key**

You can enable a primary key in Oracle using the ALTER TABLE statement.

Syntax

The syntax to enable a primary key using the ALTER TABLE statement in Oracle/PLSQL is:

ALTER TABLE table\_name

ENABLE CONSTRAINT constraint\_name;

Example

Let's look at an example of how to enable a primary key using the ALTER TABLE statement in Oracle.

ALTER TABLE supplier

ENABLE CONSTRAINT supplier\_pk;

In this example, we're enabling a primary key on the supplier table called supplier\_pk.

## What is a foreign key in Oracle?

A foreign key is a way to enforce referential integrity within your Oracle database. A foreign key means that values in one table must also appear in another table.

The referenced table is called the parent table while the table with the foreign key is called the child table. The foreign key in the child table will generally reference a [primary key](https://www.techonthenet.com/oracle/primary_keys.php) in the parent table.

A foreign key can be defined in either a CREATE TABLE statement or an ALTER TABLE statement.

## Using a CREATE TABLE statement

### Syntax

The syntax for creating a foreign key using a CREATE TABLE statement is:

CREATE TABLE table\_name

(

column1 datatype ,

column2 datatype ,

...

CONSTRAINT fk\_column

FOREIGN KEY (column1, column2, ... column\_n)

REFERENCES parent\_table (column1, column2, ... column\_n)

);

### Example

CREATE TABLE supplier

( supplier\_id number (10) ,

supplier\_name varchar2(50) not null,

address varchar2(50),

CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id)

);

CREATE TABLE products

( product\_id number (10) not null,

supplier\_id number (10) not null,

CONSTRAINT fk\_supplier

FOREIGN KEY (supplier\_id)

REFERENCES supplier(supplier\_id)

);

CREATE TABLE supplier

( supplier\_id numeric(10) not null,

supplier\_name varchar2(50) not null,

contact\_name varchar2(50),

CONSTRAINT supplier\_pk PRIMARY KEY (supplier\_id, supplier\_name)

);

CREATE TABLE products

( product\_id numeric(10) not null,

supplier\_id numeric(10) not null,

supplier\_name varchar2(50) not null,

CONSTRAINT fk\_supplier\_comp

FOREIGN KEY (supplier\_id, supplier\_name)

REFERENCES supplier(supplier\_id, supplier\_name)

);

In this example, our foreign key called fk\_foreign\_comp references the supplier table based on two fields - the supplier\_id and supplier\_name fields.

## Using an ALTER TABLE statement

### Syntax

The syntax for creating a foreign key in an ALTER TABLE statement is:

ALTER TABLE table\_name

ADD CONSTRAINT constraint\_name

FOREIGN KEY (column1, column2, ... column\_n)

REFERENCES parent\_table (column1, column2, ... column\_n);

### Example

ALTER TABLE products

ADD CONSTRAINT fk\_supplier

FOREIGN KEY (supplier\_id)

REFERENCES supplier(supplier\_id);

In this example, we've created a foreign key called fk\_supplier that references the supplier table based on the supplier\_id field.

We could also create a foreign key with more than one field as in the example below:

ALTER TABLE products

ADD CONSTRAINT fk\_supplier

FOREIGN KEY (supplier\_id, supplier\_name)

REFERENCES supplier(supplier\_id, supplier\_name);

Assignment:

* create table dept1(
* deptno number(2,0),
* dname varchar2(14),
* loc varchar2(13),
* constraint pk\_dept primary key (deptno)
* )

Table created.

* Statement2

Create the EMP1 table which has a foreign key reference to the DEPT table. The foreign key will require that the DEPTNO in the EMP table exist in the DEPTNO column in the DEPT table.

create table emp1(

empno number(4,0),

ename varchar2(10),

job varchar2(9),

mgr number(4,0),

hiredate date,

sal number(7,2),

comm number(7,2),

deptno number(2,0),

constraint pk\_emp primary key (empno),

constraint fk\_deptno foreign key (deptno) references dept1 (deptno)

)

* 3

Insert row into DEPT table using named columns.

insert into DEPT (DEPTNO, DNAME, LOC)

values(10, 'ACCOUNTING', 'NEW YORK')

1 row(s) inserted.

* Statement4

Insert a row into DEPT table by column position.

insert into dept

values(20, 'RESEARCH', 'DALLAS')

1 row(s) inserted.

* Statement5
* insert into dept

values(30, 'SALES', 'CHICAGO')

1 row(s) inserted.

* Statement6
* insert into dept

values(40, 'OPERATIONS', 'BOSTON')

1 row(s) inserted.

* Statement7

Insert EMP row, using TO\_DATE function to cast string literal into an oracle DATE format.

insert into emp1

values(

7839, 'KING', 'PRESIDENT', null,

to\_date('17-11-1981','dd-mm-yyyy'),

5000, null, 10

)

1 row(s) inserted.

* Statement8
* insert into emp
* values(
* 7698, 'BLAKE', 'MANAGER', 7839,
* to\_date('1-5-1981','dd-mm-yyyy'),
* 2850, null, 30

)

1 row(s) inserted.

* Statement9
* insert into emp
* values(
* 7782, 'CLARK', 'MANAGER', 7839,
* to\_date('9-6-1981','dd-mm-yyyy'),
* 2450, null, 10

)

1 row(s) inserted.

* Statement10
* insert into emp
* values(
* 7566, 'JONES', 'MANAGER', 7839,
* to\_date('2-4-1981','dd-mm-yyyy'),
* 2975, null, 20

)

1 row(s) inserted.

* Statement11
* insert into emp
* values(
* 7788, 'SCOTT', 'ANALYST', 7566,
* to\_date('13-JUL-87','dd-mm-rr') - 85,
* 3000, null, 20

)

1 row(s) inserted.

* Statement12
* insert into emp
* values(
* 7902, 'FORD', 'ANALYST', 7566,
* to\_date('3-12-1981','dd-mm-yyyy'),
* 3000, null, 20

)

1 row(s) inserted.

* Statement13
* insert into emp
* values(
* 7369, 'SMITH', 'CLERK', 7902,
* to\_date('17-12-1980','dd-mm-yyyy'),
* 800, null, 20

)

1 row(s) inserted.

* Statement14
* insert into emp
* values(
* 7499, 'ALLEN', 'SALESMAN', 7698,
* to\_date('20-2-1981','dd-mm-yyyy'),
* 1600, 300, 30

)

1 row(s) inserted.

* Statement15
* insert into emp
* values(
* 7521, 'WARD', 'SALESMAN', 7698,
* to\_date('22-2-1981','dd-mm-yyyy'),
* 1250, 500, 30

)

1 row(s) inserted.

* Statement16
* insert into emp
* values(
* 7654, 'MARTIN', 'SALESMAN', 7698,
* to\_date('28-9-1981','dd-mm-yyyy'),
* 1250, 1400, 30

)

1 row(s) inserted.

* Statement17
* insert into emp
* values(
* 7844, 'TURNER', 'SALESMAN', 7698,
* to\_date('8-9-1981','dd-mm-yyyy'),
* 1500, 0, 30

)

1 row(s) inserted.

* Statement18
* insert into emp
* values(
* 7876, 'ADAMS', 'CLERK', 7788,
* to\_date('13-JUL-87', 'dd-mm-rr') - 51,
* 1100, null, 20

)

1 row(s) inserted.

* Statement19
* insert into emp
* values(
* 7900, 'JAMES', 'CLERK', 7698,
* to\_date('3-12-1981','dd-mm-yyyy'),
* 950, null, 30

)

1 row(s) inserted.

* Statement20
* insert into emp
* values(
* 7934, 'MILLER', 'CLERK', 7782,
* to\_date('23-1-1982','dd-mm-yyyy'),
* 1300, null, 10

)

1 row(s) inserted.

* Statement21

Simple natural join between DEPT and EMP tables based on the primary key of the DEPT table DEPTNO, and the DEPTNO foreign key in the EMP table.

select ename, dname, job, empno, hiredate, loc

from emp, dept

where emp.deptno = dept.deptno

order by ename

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ENAME** | **DNAME** | **JOB** | **EMPNO** | **HIREDATE** | **LOC** |
| ADAMS | RESEARCH | CLERK | 7876 | 23-MAY-87 | DALLAS |
| ALLEN | SALES | SALESMAN | 7499 | 20-FEB-81 | CHICAGO |
| BLAKE | SALES | MANAGER | 7698 | 01-MAY-81 | CHICAGO |
| CLARK | ACCOUNTING | MANAGER | 7782 | 09-JUN-81 | NEW YORK |
| FORD | RESEARCH | ANALYST | 7902 | 03-DEC-81 | DALLAS |
| JAMES | SALES | CLERK | 7900 | 03-DEC-81 | CHICAGO |
| JONES | RESEARCH | MANAGER | 7566 | 02-APR-81 | DALLAS |
| KING | ACCOUNTING | PRESIDENT | 7839 | 17-NOV-81 | NEW YORK |
| MARTIN | SALES | SALESMAN | 7654 | 28-SEP-81 | CHICAGO |
| MILLER | ACCOUNTING | CLERK | 7934 | 23-JAN-82 | NEW YORK |
| SCOTT | RESEARCH | ANALYST | 7788 | 19-APR-87 | DALLAS |
| SMITH | RESEARCH | CLERK | 7369 | 17-DEC-80 | DALLAS |
| TURNER | SALES | SALESMAN | 7844 | 08-SEP-81 | CHICAGO |
| WARD | SALES | SALESMAN | 7521 | 22-FEB-81 | CHICAGO |

14 rows selected.

* Statement22

The GROUP BY clause in the SQL statement allows aggregate functions of non grouped columns. The join is an inner join thus departments with no employees are not displayed.

SQL-QUERIES

1. Display all the information of the EMP table?

2. Display unique Jobs from EMP table?

3. List the emps in the asc order of their Salaries?

4. List the details of the emps in asc order of the Dptnos and desc of Jobs?;

5. Display all the unique job groups in the descending order?

6. List the emps who joined before 1981.);

7. List the Empno, Ename, Sal, Daily sal of all emps in the asc order of sal ;

8. List the Empno, Ename, Sal, Exp of all emps working for Mgr 7369.

9. Display all the details of the emps whose Comm. Is more than their Sal..

10. List the emps along with their dept and Daily Sal is more than Rs.100.

11. List the emps who are either ‘CLERK’ or ‘ANALYST’ in the Desc order.

12. List the emps who joined on 1-MAY-81,3-DEC-81,17-DEC-81,19-JAN-80

13. List the emp who are working for the Deptno 10 or 20.

List the emps who joined on 1-MAY-81,3-DEC-81,17-DEC-81,19-JAN-80 in asc order of seniority.

List the emps who are joined in the year 81.

List the emps Who Annual sal ranging from 22000 and 45000.

List the Enames those are starting with ‘S’

List the emps who does not belong to Deptno 20.

List the emps those are having four chars and third character must be ‘r’.

List all the emps except ‘PRESIDENT’ & ‘MGR” in asc order of Salaries.

Set operators

Set operators are used to join the results of two (or more) SELECT statements.The SET operators available in Oracle 11g are UNION,UNION ALL,INTERSECT,and MINUS.

The UNION set operator returns the combined results of the two SELECT statements.Essentially,it removes duplicates from the results i.e. only one row will be listed for each duplicated result.To counter this behavior,use the UNION ALL set operator which retains the duplicates in the final result.INTERSECT lists only records that are common to both the SELECT queries; the MINUS set operator removes the second query's results from the output if they are also found in the first query's results. INTERSECT and MINUS set operations produce unduplicated results.

All the SET operators share the same degree of precedence among them.Instead,during query execution, Oracle starts evaluation from left to right or from top to bottom.If explicitly parentheses are used, then the order may differ as parentheses would be given priority over dangling operators.

## Points to remember -

* Same number of columns must be selected by all participating SELECT statements.Column names used in the display are taken from the first query.
* Data types of the column list must be compatible/implicitly convertible by oracle. Oracle will not perform implicit type conversion if corresponding columns in the component queries belong to different data type groups.For example, if a column in the first component query is of data type DATE, and the corresponding column in the second component query is of data type CHAR,Oracle will not perform implicit conversion, but raise ORA-01790 error.
* Positional ordering must be used to sort the result set. Individual result set ordering is not allowed with Set operators. ORDER BY can appear once at the end of the query. For example,
* UNION and INTERSECT operators are commutative, i.e. the order of queries is not important; it doesn't change the final result.
* Performance wise, UNION ALL shows better performance as compared to UNION because resources are not wasted in filtering duplicates and sorting the result set.
* Set operators can be the part of sub queries.

SELECT supplier\_id

FROM suppliers

UNION

SELECT supplier\_id

FROM products;

Functions in oracle

There are two types of functions in Oracle.  
**1) Single Row Functions:** Single row or Scalar functions return a value for every row that is processed in a query.  
**2) Group Functions:**These functions group the rows of data based on the values returned by the query. This is discussed in SQL GROUP Functions. The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

Oracle aggregate functions calculate on a group of rows and return a single value for each group.

We commonly use the aggregate functions together with the [GROUP BY](https://www.oracletutorial.com/oracle-basics/oracle-group-by/) clause. The GROUP BY clause divides the rows into groups and an aggregate function calculates and returns a single result for each group.

If you use aggregate functions without a GROUP BY clause, then the aggregate functions apply to all rows of the queried tables or [views](https://www.oracletutorial.com/oracle-view/).

We also use the aggregate functions in the [HAVING](https://www.oracletutorial.com/oracle-basics/oracle-having/) clause to filter groups from the output based on the results of the aggregate function

## The SQL COUNT(), AVG() and SUM() Functions

The COUNT() function returns the number of rows that matches a specified criterion.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

### **COUNT FUNCTION**

* COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
* COUNT function uses the COUNT(\*) that returns the count of all the rows in a specified table. COUNT(\*) considers duplicate and Null.
* COUNT(\*)
* or
* COUNT( [ALL|DISTINCT] expression )

SELECT COUNT(column\_name)  
FROM table\_name  
WHERE condition;

SELECT COUNT (\*) FROM emp  
WHERE deptno = 10;

SELECT COUNT (DISTINCT name) FROM employee;

SELECT COUNT(\*)

FROM PRODUCT\_MAST;

WHERE RATE>=20;

SELECT COUNT(empno), job

FROM emp

GROUP BY job

ORDER BY count (empno) DESC

### **AVG function**

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

**Syntax**

1. AVG()
2. or
3. AVG( [ALL|DISTINCT] expression )

**Example:**

1. SELECT AVG(COST)
2. FROM PRODUCT\_MAST;

SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

### **SUM Function**

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

**Syntax**

1. SUM()
2. or
3. SUM( [ALL|DISTINCT] expression )

SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

SELECT location, dept, SUM (salary)  
FROM employee  
GROUP BY location, dept;

SELECT empno, SUM (SAl)

FROM emp

GROUP BY empno

HAVING SUM (SAL > 10000;

### **MAX Function**

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

**Syntax**

1. MAX()
2. or
3. MAX( [ALL|DISTINCT] expression )

**Example:**

1. SELECT MAX(RATE)
2. FROM PRODUCT\_MAST;

SELECT deptno, Max(sal)

FROM emp

GROUP BY deptno

### **MIN Function**

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

**Syntax**

1. MIN()
2. or
3. MIN( [ALL|DISTINCT] expression )

**Example:**

1. SELECT MIN(RATE)
2. FROM PRODUCT\_MAST;

SELECT deptno, MIN(sal)

FROM emp

GROUP BY deptno

/

The HAVING clause is a conditional option that is directly related to the GROUP BY clause option because a HAVING clause eliminates rows from a result table based on the result of a GROUP BY clause.

SELECT department\_id, AVG(Salary)

FROM employees

HAVING AVG(Salary) > 33000;

ERROR at line 1: ORA-00937: not a single-group group function

SELECT SUBJECT, YEAR, Count(\*)

FROM Student

GROUP BY SUBJECT, YEAR;

**Character Functions**

Character functions operate on values of dataype  CHAR, VARCHAR and VARCHAR2

**LOWER**

Returns a given string in lower case.

select LOWER(‘SAMI’) from dual;  
  
LOWER  
-------------  
sami

**UPPER**

Returns a given string in UPPER case.

select UPPER(‘Sami’) from dual;  
  
UPPER  
------------------  
SAMI

**INITCAP**

Returns a given string with Initial letter in capital.

select INITCAP(‘mohammed sami’) from dual;  
  
INITCAP  
------------------  
Mohammed Sami

**LENGTH**

Returns the length of a given string.

select length(‘mohammed sami’) from dual;  
  
LENGTH  
------------  
        13

**SUBSTR**

Returns a substring from a given string. Starting from position p to n characters.

For example the following query returns “sam” from the string “mohammed sami”.

select substr('mohammed sami',10,3) from dual;  
  
Substr  
--------  
sam

**INSTR**

Tests whether a given character occurs in the given string or not. If the character occurs in the string then returns the first position of its occurrence otherwise returns 0.

Example

The following query tests whether the character “a” occurs in string “mohammed sami”

select instr('mohammed sami','a') from dual;  
  
INSTR  
--------  
4

**REPLACE**

Replaces a given set of characters in a string with another set of characters.

Example

The following query replaces “mohd” with “mohammed” .

select replace('ali mohd khan','mohd','mohammed') from dual;  
  
REPLACE  
---------  
ali mohammed khan

**RPAD**

Right pads a given string with a given character to n number of characters.

Example

The following query rights pad ename with '\*'  until it becomes 10 characters.

select rpad(ename,'\*',10) from emp;  
  
Ename  
----------  
Smith\*\*\*\*\*  
John\*\*\*\*\*\*  
Mohammed\*\*  
Sami\*\*\*\*\*\*

**LPAD**

Left pads a given string with a given character upto n number of characters.

Example

The following query left pads ename with '\*'  until it becomes 10 characters.

select lpad(ename,'\*',10) from emp;  
  
Ename  
----------  
\*\*\*\*\*Smith  
\*\*\*\*\*\*John  
\*\*Mohammed  
\*\*\*\*\*\*Sami

**LTRIM**

Trims blank spaces from a given string from left.

Example

The following query returns string “       Interface        “ left trimmed.

select ltrim('       Interface       ') from dual;  
  
Ltrim  
--------------  
Interface

**RTRIM**

Trims blank spaces from a given string from Right.

Example

The following query returns string “       Interface        “ right trimmed.

select rtrim('       Interface       ') from dual;  
  
Rtrim  
------------  
   Interface

**TRIM**

Trims a given character from left or right or both from a given string.

Example

The following query removes zero from left and right of a given string.

Select trim(0 from '00003443500') from dual;  
  
Trim  
----------  
34435

**CONCAT**

Combines a given string with another string.

Example

The following Query combines ename with literal string “ is a “ and jobid.

Select concat(concat(ename,' is a '),job) from emp;  
  
Concat  
----------------  
Smith is a clerk  
John is a Manager  
Sami is a G.Manager