**SQL Learning Material**

1. **Create tables and specify the queries in SQL**.

**Basic Data Types:-** Data types come in several forms and sizes, allowing the programmer to create tables suited to the scope of the project, The decision made in choosing proper data types greatly influence the performance of a database, so it is wise to have a detailed understanding of these data types.

Oracle is capable of many of the data types that even the novice programmer has probably already been exposed to. Some of the most commonly used data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| CHAR(size) | This data type is used to store character strings values of fixed length. The size in the brackets determines the number of characters it can hold. The maximum number of characters (i.e. the size) this data type can hold is 255 characters. The data held is right padded with spaces to whatever length specified. |
| VARCHAR(size)/ VARCHAR2(size) | This data type is used to store variable length alphanumeric data. It is a flexible form of CHAR data type. The maximum data this data type can hold is 4000 characters. Inserted values are not padded. |
| DATE | This data type is used to represent date and time. To enter dates other than the standard format, use the appropriate functions. Date time stores date in the 24-hour format. Valid dates range from January 1,4212 B.C. to December 31,4712 A.D. |
| NUMBER(P,S) | The NUMBER data type is used to store numbers(fixed or floating point). Numbers of virtually any magnitude maybe stored up to 38 digits of precision. Valid values are 0 and positive and negative numbers with magnitude 1.0E-130 to 9.9…E125. The precision (P) determines the maximum length of the data, whereas the scale (S) ,determines the number of places to the right of the decimal. If scale is omitted then the default is zero. If the precision is omitted, values are stored with their original precision upto the maximum of 38 digits |
| LONG | This data type is used to variable length character strings containing upto 2GB. LONG data can be used to store arrays of binary data in ASCII format. Only one long value can be defined per table. A table containing LONG values cannot be clustered. |
| RAW /  LONG RAW | The RAW / LONG RAW data types are used to store binary data, such as digitized picture or image. Data loaded into columns of these data types are stored without any further conversion. RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to 2GB. Values stored in columns having LONG RAW data type cannot be indexed. |

1. **Creating A Table**

**Command:-** Create Table

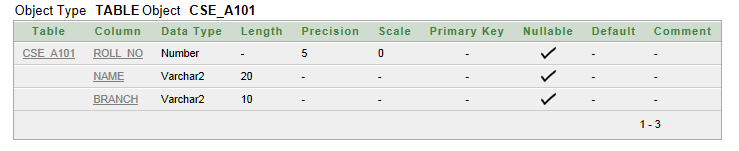
**Purpose:-** The Create Table command is used to create a table in the database. It consists of rows and columns. Each column has a minimum of three attributes- a name, a data type, and size(i.e. column width).

**Syntax:-** Create table <TableName> (<ColumnName1> <Data Type>(<Size>), <ColumnName2> <Data Type>(<Size>), ………..<ColumnName n> <Data Type>(<Size>));

**Example:-** Create table STUDENT (Roll\_no number(5), Name varchar2(20), Branch varchar2(10));

Desc STUDENT;

**Output:-**



1. **Inserting Data Into The Table**

**Command:-** Insert Into

**Purpose:-** The Insert Into Table command is used to load the created table with data to be manipulated later.

**Syntax:-** Insert into <table name>(<col1>,<col2>,……..,<col n>) values (<expression1>, <expression2> ,……,<expression3>);

**Example:-**

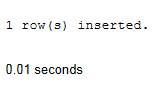
Insert into STUDENT values(101, ‘Aanchal’, ’CSE’);

Insert into STUDENT values(126, ‘Jotnain’, ’CSE’);

Insert into STUDENT values(149, ‘Sapanpreet’, ’ECE’);

Insert into STUDENT values(151, ‘Savita’, ’CSE’);

**Output:-**



1. **Viewing Data In The Tables**

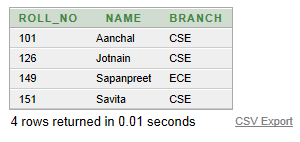
**Command:-** Selecting all rows and all columns

**Purpose:-** This command is used to view all the rows and columns of the table created in the database.

**Syntax:-** Select \* from <table name>;

**Example:-** Select \* from STUDENT;

**Output:-**



1. **Filtering Table Data**
2. Selected rows and all columns

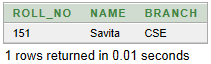
**Command:-** Select

**Purpose:-** If the information of a particular student is to be retrieved from a table, its retrieval must be based on a specific condition

**Syntax:-** Select \* from <table name> where <condition>;

**Example:-** Select \* from STUDENT where roll\_no=151;

**Output:-**



1. Selected columns and all rows

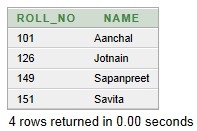
**Command:-** Select

**Purpose:-** To filter the data that is not required

**Syntax:-** Select <col name1>, <col name2> from <table name>;

**Example:-** Select Roll\_no, name from STUDENT;

**Output:-**



1. Selected columns and selected rows

**Command:-** Select

**Purpose:-** To filter the data that is not required

**Syntax:-** Select <col name1>, <col name2> from <table name> where <condition>;

**Example:-** Select roll\_no, name from STUDENT where name=’Aanchal’;

**Output:-**



1. **Eliminating Duplicate Rows When Using A Select Statement**

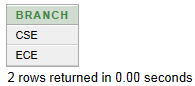
**Command:-** Distinct

**Purpose:-** A table could hold duplicate rows from display. This could be removed by using DISTINCT clause

**Syntax:-** Select Distinct <col name1>, <col name2> from <table name>;

**Example:-** Select distinct branch from STUDENT;

**Output:-**



1. **Sorting Data In A Table**

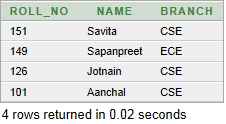
**Command:-** Order By

**Purpose:-** Oracle allows data from a table to be viewed in a sorted order. The rows retrieved from the table will be sorted in either **ascending** or **descending** order.

**Syntax:-** Select \* from <table name> order by <col name1>, <col name2> <[Sort order]>;

**Example:-** Select \* from STUDENT order by roll\_no desc;

**Output:-**



1. **Creating A Table From A Table**

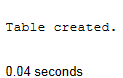
**Command:-** Create table …..as Select

**Purpose:-** To create a copy of existing table with create table and select statements.

**Syntax:-** Create Table <TableName> (<ColumnName1>, <ColumnName2>,…) as select <colname1>, <colname2>,…. From <existing table name>;

**Example:-** Create table CSEA\_Data101(rollno, Name) as select roll\_no, name from STUDENT;

**Output:-**



1. **Inserting Data Into A Table From Another Table**

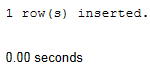
**Command:-** Insert into….Select

**Purpose:-** To filter the data that is not required

**Syntax:-** Insert into <table name>Select <col name1>, <col name2>…. from <table name>;

**Example:-** Insert into CSEA\_Data101 Select roll\_no,name from STUDENT where roll\_no=149;

**Output:-**



1. **Delete Operations**

**Command:-** Delete

**Purpose:-** To delete the rows from the table that satisfies the condition provided by its where clause and returns the number of records deleted.

**Syntax:-**

* Removal of all rows

Delete from <table name>;

* Removal of Specific Row(s)

Delete from <table name> where <condition>;

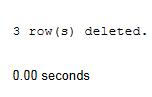
**Example:-**

Delete from CSEA\_Data101 where rollno=149;

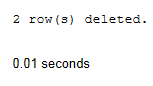
Delete from CSEA\_Data101;

**Output:-**

* All Rows



* Specific Row(s)



1. **Updating The Contents Of A Table**

**Command:-** Update……set

**Purpose:-** The Update statement updates columns in the existing table’s rows with new values. The Set clause indicates which column data should be modified and the new valued they should hold.

**Syntax:-**

* Updating Specific Records

Update <table name> set <col name1>=<expression1>, <col name2>=<expression2> where <condition>;

* Updating All The Records

Update <table name> set <col name1>=<expression1>, <col name2>=<expression2>;

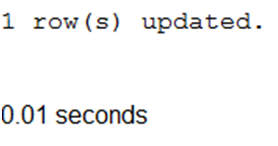
**Example:-**

Update STUDENT set Branch=’ECE’ where roll\_no=101;

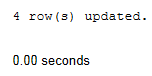
Update STUDENT set Branch=’CSE’;

**Output:-**

* Updating Specific Records



* Updating All The Records



1. **Modifying The Structure Of A Table**

**Command:-** Alter Table

**Purpose:-** The structure of a table can be modified by using a alter table command. Alter Table allows changing the structure of an existing table. With Alter Table it is possible to add or delete columns, create or destroy indexes, change the data type of existing columns, or rename columns or the table itself

**Syntax:-**

* Adding New Columns

Alter table <table name> add(<new col name> <data type> (<Size>), <new col name> <data type> (<Size>),…..);

* Modifying Existing Column

Alter table <table name> modify(<col name> <new data type> (< new Size>), <col name> <new data type> (<new Size>),…..);

* Dropping A Column

Alter table <table name> drop column <col name>;

**Example:-**

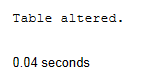
Alter table STUDENT add( Telephone number(10));

Alter table STUDENT modify( Telephone number(20));

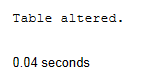
Alter table STUDENT drop column Telephone;

**Output:-**

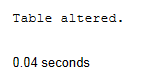
* Adding a new column



* Modifying Existing Column



* Dropping A Cloumn



1. **Renaming Tables**

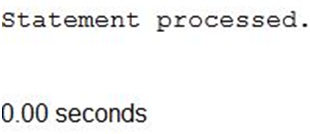
**Command:-** Rename

**Purpose:-** To change the name of a table as per the user’s wish or requirement.

**Syntax:-** Rename <new table name> to <old table name>;

**Example:-** Rename STUDENT to group\_a101;

**Output:-**



1. **Truncating Tables**

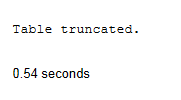
**Command:-** Truncate Table

**Purpose:-** Truncate Tables empties the table completely. Once the data deleted cannot be retrieved.

**Syntax:-** Truncate Table <table name>;

**Example:-** Truncate table CSEA\_Data101;

**Output:-**



1. **Destroying Tables**

**Command:-** Drop Table

**Purpose:-** Sometimes table in the database becomes obsolete and need to be discarded, so we use drop table command.

**Syntax:-** Drop table <table name>;

**Example:-**  Drop table CSEA\_Data101;

**Output:-**

Table dropped.

0.54 seconds

1. **Implementing DDL and DML commands of SQL**
2. **Command:-** Create Table

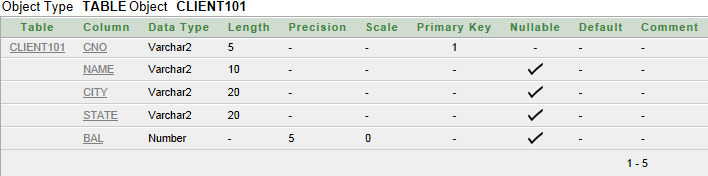
**Purpose:-** The Create Table command is used to create a table in the database.

**Syntax:-** Create table <TableName> (<ColumnName1> <Data Type>(<Size>), <ColumnName2> <Data Type>(<Size>), ………..<ColumnName n> <Data Type>(<Size>));

**Example:-** Create table Instructor (CNo varchar2(5) Primary key, Name varchar2(10), City varchar2(20), State varchar2(20), Bal number(5));

Desc instructor;

**Output:-**



1. **Command:-** Inserting Data Into The Tables

**Purpose:-** The Insert Into Table command is used to load the created table with data to be manipulated later.

**Syntax:-** Insert into <tablename>(<col1>,<col2>,……..,<col n>) values (<expression1>, <expression2> ,……,<expression3>);

**Example:-**

Insert into Instructor values('C1','Ivan','Mumbai','Maharashtra',3000);

Insert into Instructor values('C2','Mamta','Madras','TamilNadu',4000);

Insert into Instructor values('C3','Chaya','Manglore','Karnataka',3200);

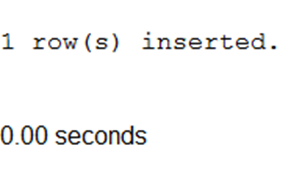
Insert into Instructor values('C4','Ashwini','Delhi','Delhi',2900);

Insert into Instructor values('C5','Yogesh','Banglore','Karnataka',5100);

Insert into Instructor values('C6','Deepak','Chandigarh','Chandigarh',0);

**Output:-**

For each of the above Insert Into statements:



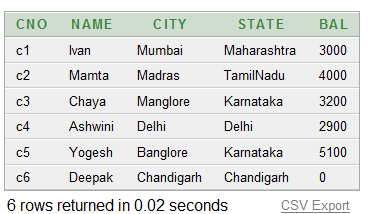
1. **Command:-**Viewing Data In The Tables.

**Purpose:-** All rows and all columns

**Syntax:-** Select \* from <table name>;

**Example:-** Select \* from Instructor;

**Output:-**



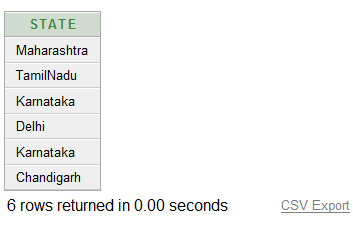
1. **Command:-** Filtering Table Data i.e., selected columns and all rows

**Purpose:-** To filter the data that is not required

**Syntax:-** Select <col name1>, <col name2> from <table name>;

**Example:-** Select state from Instructor;

**Output:**



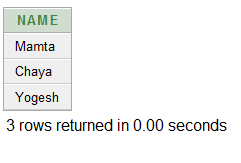
1. **Command:-** Filtering Table Data i.e., selected rows and selected columns

**Purpose:-** To limit the columns as well as rows

**Syntax:-** Select <col name1>,< col name2>,…..,<col name n> from <table name> where <condition>;

**Example:-** Select name from Instructor where bal>3000;

**Output:-**



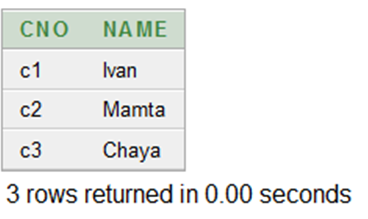
1. **Command:-** Pattern Matching

**Purpose:-** The use of ‘LIKE’ predicate to retrieve the data that matches a particular pattern.

**Syntax:-** Select <col name1>,<col name2> from <table name> where <col name> like’M%’;

**Example:-** Select cno, name from Instructor where city like ‘M%’;

**Output:-**



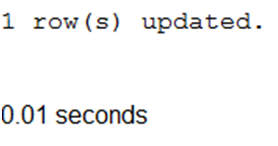
1. **Command:-** Update The Contents Of A Table.

**Purpose:-** To update records conditionally

**Syntax:-** Update <table name> set <col name1>=<expression1>, <col name2>=<expression2> where <condition>;

**Example:-**  Update Instructor set city=’Delhi’ where cno=’C6’;

**Output:-**



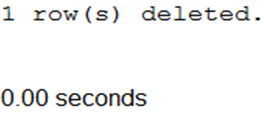
1. **Command:-** Delete Operation

**Purpose:-** To remove some specific records.

**Syntax:-** Delete From <TableName> where <condition>;

**Example:-** Delete from Instructor where bal=0;

**Output:-**



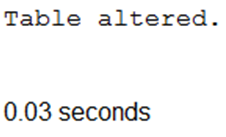
1. **Command:-** Modifying The Structure Of The Table

**Purpose:-** To add new column

**Syntax:-** Alter table <table name> add(<col name> <data type> (<Size>), <col name> <data type> (<Size>),…..);

**Example:-** Alter table Instructor add( Telephone number(10));

**Output:-**



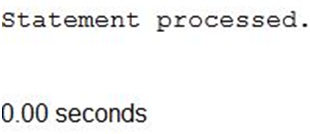
1. **Command:-** Renaming Tables

**Purpose:-** To change the name of a table as per the user’s wish or requirement.

**Syntax:-** Rename <new table name> to <old table name>;

**Example:-** Rename Instructor to Instructor1;

**Output:-**



1. **Command:-** Deleting Duplicate Rows

**Purpose:-** Eliminating duplicate rows when using a select statement.

**Syntax:-** Select distinct <col name1>,<col name2> from <table name>;

**Example:-** Select distinct city from Instructor;

**Output:-**



1. **Command:-** Aggregate Function--SUM

**Purpose:-** This is used to do the sum of the numeric values of a specific column.

**Syntax:-** sum([<distinct>/ <all>]<col name>)

**Example:-** Select sum(bal) as Total\_Bal from Instructor;

**Output:-**



1. **Command:-** Aggregate Function-- COUNT

**Purpose:-** This is used to count the number of rows in the table, including duplicates and those with nulls.

**Syntax:-** count(\*)

**Example:-** Select count(\*) “No.OfRecords” from client\_master101;

**Output:-**



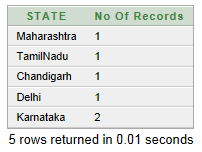
1. **Command:-** Grouping Data From Tables in SQL

**Purpose:-** It tells Oracle to group rows or records based on distinct values that exist for specified columns.

**Syntax:-** Select <colname1>,<colname2>, aggregate function(<expression>) from <table name> where <condition> group by <colname1>, <colname2>;

**Example:-**  Select state, count(\*)"No of Records" from Instructor group by state;

**Output:-**



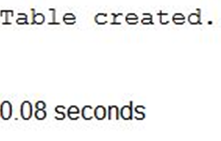
1. **Command:-** Creating A Table From A Table

**Purpose:-** To create a copy of existing table with create table and select statements.

**Syntax:-** Create Table <TableName> (<ColumnName1>, <ColumnName2>,…) as select <colname1>, <colname2>,…. From <existing table name>;

**Example:-** Create table Instructor (CNo, Name, Balance) as select cno, name, bal from client\_master101;

**Output:-**



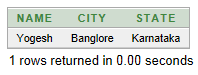
1. **Command:-** SubQueries

**Purpose:-** A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

**Syntax:-** Select <colname1>, <colname2> from <table name> where <colname> operator (select \* from <table name> where <condition>);

**Example:-** Select name, city, state from Instructor where bal=(select max(bal) from client\_master101);

**Output:-**



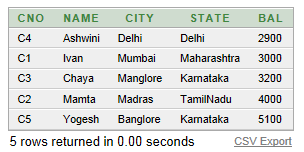
1. **Command:-** Sorting Data In A Table

**Purpose:-** This command allows data from a table to be viewed in a sorted order either in ascending order or descending order.

**Syntax:-** Select \* from <table name> order by <colname1>, <colname2> <[sort order]>;

**Example:-** Select \* from Instructor order by bal asc;

**Output:-**



1. **Command:-** Destroying The Table

**Purpose:-** Sometimes table in the database becomes obsolete and need to be discarded, so we use drop table command.

**Syntax:-** Drop table <table name>;

**Example:-**  Drop table Instructor;

**Output:-**

Table dropped.

0.54 seconds

1. **Implementing Data Constraints On The Created Table In A Database.**

**Data Constraints:-** A constraint is a limitation that you place on the data that users can enter into a column or group of columns. A constraint is part of the table definition; you can implement constraints when you create the table or later. You can remove a constraint from the table without affecting the table or the data, and you can temporarily disable certain constraints.

The distinction between column-level and table-level constraints is that you define a column-level constraint on one column only. Therefore, you can define it as part of the column definition. However, to define a constraint on more than one column—such as a primary-key constraint defined on two columns—you need to define the columns in the table first, then add the constraint definition at the end of the table definition. That constraint type is at the table level.

Following are commonly used constraints available in SQL:

* [**NOT NULL Constraint**](http://www.tutorialspoint.com/sql/sql-not-null.htm)**:** Ensures that a column cannot have NULL value.
* [**UNIQUE Constraint**](http://www.tutorialspoint.com/sql/sql-unique.htm)**:** Ensures that all values in a column are different.
* [**PRIMARY Key**](http://www.tutorialspoint.com/sql/sql-primary-key.htm)**:** Uniquely identified each rows/records in a database table.
* [**FOREIGN Key**](http://www.tutorialspoint.com/sql/sql-foreign-key.htm)**:** Uniquely identified a rows/records in any another database table.
* [**CHECK Constraint**](http://www.tutorialspoint.com/sql/sql-check.htm)**:** The CHECK constraint ensures that all values in a column satisfy certain conditions.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use ALTER TABLE statement to create constraints even after the table is created.

1. **Adding A Not Null Constraint:-**

**Command:-**  Not Null

**Purpose:-** By default all columns in a table can contain null values. If you want to ensure that a column must always have a value i.e., it should not be left blank, then define a not null constraint on it. Null value can be inserted into the columns of any data type. Not null constraint can only be applied at column level.

**Syntax:-**

* Defined at column level:

Columnname datatype(size) **NOTNULL**

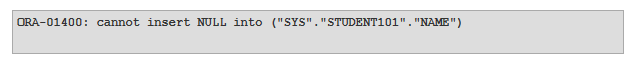
**Example:-**

* At column level:

create table STUDENT(rollno number(10), Name varchar2(20) **not null**, Branch varchar2(5));

insert into STUDENT values(101,'','CSE');

**Output:-**



This was the output when a record with null value was tried to be inserted in the STUDENT table. This verifies the not null constraint.

1. **Adding A Unique Constraint:-**

**Command:-**  Unique

**Purpose:-** The purpose of unique key is to ensure that information in the column(s) is unique i.e., a value entered in column(s) defined in the unique constraint must not be repeated across the column(s). A table may have many unique keys. A column(s) may have null values but not the duplicate values.

**Syntax:-**

* Defined at column level:

Columnname datatype(size) **Unique**

* Defined at table level:

**Unique**(column name1, column name2,….., column name n)

**Example:-**

* At column level:

create table STUDENT(rollno number(10) **Unique** , Name varchar2(20), Branch varchar2(5));

insert into STUDENT values(101,'A','CSE');

insert into STUDENT values(101,'A','CSE');

* At table level:

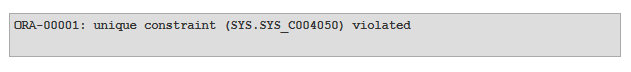
create table STUDENT(rollno number(10), Name varchar2(20), Branch varchar2(5), **Unique** (rollno, name));

insert into STUDENT values(101,'A','CSE');

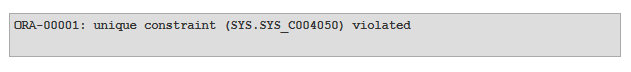
insert into STUDENT values(102,'A','CSE');

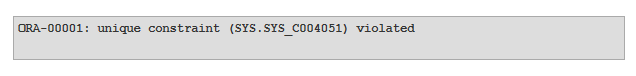
**Output:-**

* At column level



* At table level

This output came when rollno was same. It meant that you cannot enter duplicate values. This shows that unique key is verified.

This output came when rollno was different but name was same. It meant that you cannot enter duplicate values. This shows that unique key is verified.

1. **Adding A Primary Key:-**

**Command:-**  Primary Key

**Purpose:-** A primary key is one or more column(s) in a table used to uniquely identify each row in the table. A primary key column in a table has special attributes:

* It defines the column as a mandatory column i.e., the column cannot be left blank. The NOT NULL attribute is active.
* The data held across the column must be UNIQUE.

**Syntax:-**

* Defined at column level:

Columnname datatype(size) **Primary Key**

* Defined at table level:

**Primary Key**(column name1, column name2,….., column name n)

**Example:-**

* At column level:

create table STUDENT(rollno number(10) **Primary Key**, Name varchar2(20), Branch varchar2(5));

insert into STUDENT values(101,'A','CSE');

insert into STUDENT values(101,'B','CSE');

insert into STUDENT values(,'C','CSE');

* At table level:

create table stu101(ID number(10),rollno number(10) , Name varchar2(20), Branch varchar2(5), **Primary Key**(ID, rollno));

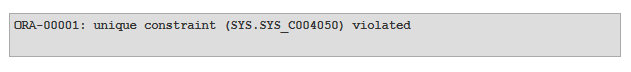
insert into stu101 values(1,101,'A','CSE');

insert into stu101 values(1,101,'B','CSE');

insert into stu101 values(,102,'C','CSE');

**Output:-**

* At column level:

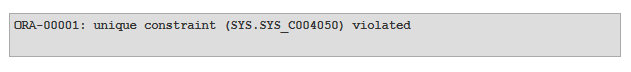


This output was recorded when duplicate value was tried to enter into the STUDENT table, which shows that primary key is applied accurately.



This output was recorded when null value was tried to enter into the STUDENT table, which shows that primary key is applied accurately.

* At table level:



This output was recorded when duplicate value was tried to enter into the stu101 table, which shows that primary key is applied accurately.



This output was recorded when null value was tried to enter into the stu101 table, which shows that primary key is applied accurately.

1. **Adding A Foreign Key:-**

**Command:-**  Foreign Key

**Purpose:-** Foreign keys represent relationships between the tables. A foreign key is a column or group of columns whose values are derived from the primary key or unique key of some other table.

The table in which the foreign key is defined is called **Foreign table** or **Detail table**. The table that defines the primary or unique key and is referred by the foreign key is called the **Primary table** or **Master table.**

The master table can be referenced in the foreign key definition by using REFERENCES adverb. If the name of the column is not specified, by default oracle uses primary key in the master table.

**Syntax:-**

* Defined at column level:

Columnname datatype(size) **References** table name [(column name)] [On Delete Cascade]

* Defined at table level:

**Foreign Key**(column name[, column name]) **References** table name [(column name [,column name])]

**Example:-**

* At column level:

create table DEPARTMENT(Dno number(10), Dname varchar2(20), Dlocation varchar2(20));

create table EMPLOYEE(Eno number(10) primary key, Ename varchar2(20), Eaddress varchar2(20), Dno number(5) **references** DEPARTMENT(dno));

insert into DEPARTMENT values(1,'Sales','Patiala');

insert into EMPLOYEE values(213,'ABC','Chandigarh',1);

insert into EMPLOYEE values(214,'DEF','Sirhind',2);

* At table level:

create table DEPARTMENT(Dno number(10), Dname varchar2(20), Dlocation varchar2(20));

create table EMPLOYEE(Eno number(10)primary key, Ename varchar2(20), Eaddress varchar2(20), Dno number(5) , **Foreign Key** (dno) **references** DEPARTMENT(dno));

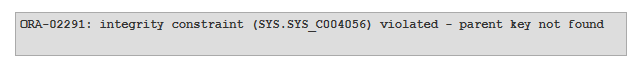
insert into DEPARTMENT values(1,'Sales','Patiala');

insert into EMPLOYEE values(213,'ABC','Chandigarh',1);

insert into EMPLOYEE values(214,'DEF','Sirhind',2);

**Output:-**

Both level of definition of foreign key has the same output.



1. **Adding A Check Constraint:-**

**Command:-**  Check

**Purpose:-** A **check constraint** is applied to each [row](http://en.wikipedia.org/wiki/Row_(database)) in the [table](http://en.wikipedia.org/wiki/Table_(database)). The constraint must be a [predicate](http://en.wikipedia.org/wiki/Predicate_(mathematical_logic)). It can refer to a single or multiple [columns](http://en.wikipedia.org/wiki/Column_(database)) of the table. The result of the predicate can be either TRUE, FALSE, or UNKNOWN, depending on the presence of [NULLs](http://en.wikipedia.org/wiki/Null_(SQL)). If the predicate evaluates to UNKNOWN, then the constraint is not violated and the row can be inserted or updated in the table. This is contrary to predicates in [WHERE](http://en.wikipedia.org/wiki/Where_(SQL)) clauses in [SELECT](http://en.wikipedia.org/wiki/Select_(SQL)) or [UPDATE](http://en.wikipedia.org/wiki/Update_(SQL)) statements.

Check constraints are used to ensure the [validity of data](http://en.wikipedia.org/wiki/Data_validation) in a database and to provide [data integrity](http://en.wikipedia.org/wiki/Data_integrity). If they are used at the database level, applications that use the database will not be able to add invalid data or modify valid data so the data becomes invalid, even if the application itself accepts invalid data.

**Syntax:-**

* Defined at column level:

Columnname datatype(size) **CHECK**(logical expression)

* Defined at table level:

**CHECK**(logical expression)

**Example:-**

* At column level:

create table EMPLOYEE(Eno number(10)primary key, Ename varchar2(20), Eaddress varchar2(20), Dno number(5), Age number(5) **CHECK** (Age>18));

insert into EMPLOYEE values(101,'DEF','Chandigarh',1,23);

insert into EMPLOYEE values(104,'ABC','Chandigarh',1,17);

* At table level:

create table STUDENT(Eno number(10)primary key, Ename varchar2(20), Eaddress varchar2(20), Dno number(5), Age number(5), **CHECK** (Age>18, address IN(‘Sirhind’,’Chandigarh’));

insert into STUDENT values(101,'DEF','Chandigarh',1,23);

insert into STUDENT values(101,'DEF','Chandigarh',1,12);

insert into STUDENT values(104,'ABC','Nabha',1,17);

**Output:-**

Both column level and table level has the same output:



This was the output recorded when records were entered that had other values as defined in the logical expression of check condition.

1. **Implement the structure of the table.**

Alter command is used for altering the table structure, such as,

* to add a column to existing table
* to rename any existing column
* to change data type of any column or to modify its size.
* to drop a column from the table.

## **1)Alter Command:** Add a new Column

Using ALTER command we can add a column to any existing table. Following is the syntax,

ALTER TABLE table\_name ADD(

column\_name datatype);

Here is an Example for this,

ALTER TABLE student ADD(

address VARCHAR(200)

);

## 2)Alter Command: Add multiple new Columns

Using  this we can even add multiple new columns to any existing table. Following is the syntax,

ALTER TABLE table\_name ADD(

column\_name1 datatype1,

column-name2 datatype2,

column-name3 datatype3);

Here is an Example for this,

ALTER TABLE student ADD(

father\_name VARCHAR(60),

mother\_name VARCHAR(60),

dob DATE);

## 3) Alter Command: Modify an existing Column

## Using Alter command can also be used to modify data type of any existing column. Following is the syntax,

ALTER TABLE table\_name modify(

column\_name datatype

);

Here is an Example for this,

ALTER TABLE student MODIFY(

address varchar(300));

.

4)Rename a Column Using  Alter command you can rename an existing column. Following is the syntax,

ALTER TABLE table\_name RENAME column old\_column\_name TO new\_column\_name;

ALTER TABLE student RENAME column

address TO location;

To change the name of the table.

Alter table iot rename to iot1;

## 5)Alter Command: Drop a Column

 Alter command can also be used to drop or remove columns. Following is the syntax,

ALTER TABLE table\_name DROP

column\_name;

Here is an example for this,

ALTER TABLE student DROP(

address);

The above command will drop the  column from the table **student**

6) Alter is used to add the constraint in the table,

Alter table books add constraint pk\_books primary key(book\_id);

7)Alter is used to drop constraint.

Alter table books drop constraint pk\_books ;