

# SQL COMMANDS

# Basic SQL Statements

---

Data Definition Language	Data Manipulation Language	Data Control Language	Transaction Control Language
Deals with database schemas and descriptions, of how the data should reside in the database.	Deals with data manipulation, and includes most common SQL statements, and it is used to store, modify, retrieve, delete and update data in database.	Includes commands such as GRANT, and mostly concerned with rights, permissions and other controls of the database system.	Deals with transaction within a database.
<ul style="list-style-type: none"><li>• Create</li><li>• Alter</li><li>• Drop</li></ul>	<ul style="list-style-type: none"><li>• Select</li><li>• Insert</li><li>• Update</li><li>• Delete</li></ul>	<ul style="list-style-type: none"><li>• Grant</li><li>• Revoke</li></ul>	<ul style="list-style-type: none"><li>• Commit</li><li>• Rollback</li><li>• Savepoint</li></ul>

## DDL – Data Definition Language

Command	Description
Create	Creates objects in the database/database objects
Alter	Alters the structures of the database/ database objects
Drop	Deletes objects from the database
Truncate	Removes all records from a table permanently
Rename	Renames an object

# Create Table Statements

The CREATE TABLE statement is used to create a new table in selected database

## Syntax:

```
CREATE TABLE <table_name> (  
    columnA datatype,  
    columnB datatype  
);
```

```
SQL> CREATE TABLE EMPLOYEE  
2   ( EMPID NUMBER(4),  
3   NAME VARCHAR(20),  
4   BIRTHDATE DATE);
```

Table created.

EMPID	NAME	BIRTHDATE

# Alter Table Statements

Alter Table is used to:

- Add, Delete, Modify columns from existing table
- Add/Drop constraints on an existing table

Syntax:

```
ALTER TABLE <table_name> ADD <column_name> <datatype>
```

```
ALTER TABLE <table_name> DROP column <column_name>
```

```
ALTER TABLE <table_name> RENAME COLUMN <old_name> TO <new_name>
```

```
ALTER TABLE <table_name> MODIFY <column_name> <datatype> [constraints];
```

```
SQL> ALTER TABLE EMPLOYEE  
2 ADD MOBILE NUMBER(10);
```

Table altered.

## EMPLOYEE

EMPID	NAME	BIRTHDATE	MOBILE

```
SQL> ALTER TABLE EMPLOYEE  
2 RENAME COLUMN MOBILE TO CONTACT;
```

Table altered.

## EMPLOYEE

EMPID	NAME	BIRTHDATE	CONTACT

# Alter Table Statements

```
SQL> ALTER TABLE EMPLOYEE  
2 MODIFY CONTACT NUMBER(12);
```

Table altered.

```
SQL> DESC EMPLOYEE
```

Name	Null?	Type
EMPID		NUMBER(4)
NAME		VARCHAR2(20)
BIRTHDATE		DATE
CONTACT		NUMBER(12)

**EMPLOYEE**

EMPID	NAME	BIRTHDATE	CONTACT

```
SQL> ALTER TABLE EMPLOYEE  
2 DROP COLUMN CONTACT  
3 ;
```

Table altered.

```
SQL> DESC EMPLOYEE
```

Name	Null?	Type
EMPID		NUMBER(4)
NAME		VARCHAR2(20)
BIRTHDATE		DATE

**EMPLOYEE**

EMPID	NAME	BIRTHDATE

# SQL Constraints

Constraint	Description
Not Null	Ensures that a column does not have a NULL value.
Default	Provides a default value for a column when none is specified.
Unique	Ensures that all the values in a column are different.
Primary	Identifies each row/record in a database table uniquely.
Check	Ensures that all values in a column satisfy certain conditions.
Index	Creates and retrieves data from the database very quickly.

# SQL Constraints

```
CREATE TABLE <table_name>
(
  column1 <datatype> [constraints],
  column2 <datatype> [constraints],
  column3 <datatype> [constraints],
  ....
);
```

```
SQL> CREATE TABLE CUSTOMER
2  ( ID NUMBER(4) NOT NULL UNIQUE,
3   NAME VARCHAR(20) NOT NULL,
4   AGE INT NOT NULL CHECK (AGE>=18));
_
Table created.
```

```
SQL> CREATE TABLE CUSTOMER
2  ( ID NUMBER(4) NOT NULL PRIMARY KEY,
3   NAME VARCHAR(20),
4   AGE INT DEFAULT 13
5   );
Table created.
```



```
SQL> ALTER TABLE CUSTOMER  
2 MODIFY AGE INT NOT NULL;
```

Table altered.

```
SQL> DESC CUSTOMER
```

Name	Null?	Type
-----	-----	-----
ID	NOT NULL	NUMBER(4)
NAME		VARCHAR2(20)
AGE	NOT NULL	NUMBER(38)

Similarly, CHECK and PRIMARY KEY can be added using ALTER ADD SQL command.

```
SQL> ALTER TABLE CUSTOMER  
2 ADD UNIQUE (NAME);
```

Table altered.

The VALUE of the primary key is made up of TWO COLUMNS (ID + LastName).

```
SQL> CREATE TABLE Persons (  
2 ID int NOT NULL,  
3 LastName varchar(255) NOT NULL,  
4 FirstName varchar(255),  
5 Age int,  
6 CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)  
7 );
```

Table created.

# Rename Table Statement

---

## SYNTAX:

**RENAME** <table\_name> **TO** <new\_name>;

```
SQL> rename Persons to emp;
```

```
Table renamed.
```

```
SQL> desc emp
```

Name	Null?	Type
-----		
ID	NOT NULL	NUMBER(38)
LASTNAME	NOT NULL	VARCHAR2(255)
FIRSTNAME		VARCHAR2(255)
AGE		NUMBER(38)

# Truncate Table Statement

---

## SYNTAX:

```
TRUNCATE TABLE <table_name>;
```

Table renamed.

```
SQL> desc emp
```

Name	Null?	Type
ID	NOT NULL	NUMBER(38)
LASTNAME	NOT NULL	VARCHAR2(255)
FIRSTNAME		VARCHAR2(255)
AGE		NUMBER(38)

```
SQL> TRUNCATE TABLE EMP;
```

Table truncated.

# Drop Table Statements

The DROP TABLE statement is used to drop a table from selected database

Syntax:

```
DROP TABLE <table_name>;
```

Example:

```
DROP TABLE employee;
```

```
SQL> TRUNCATE TABLE EMP;
```

Table truncated.

```
SQL> DESC EMP
```

Name	Null?	Type
ID	NOT NULL	NUMBER(38)
LASTNAME	NOT NULL	VARCHAR2(255)
FIRSTNAME		VARCHAR2(255)
AGE		NUMBER(38)

```
SQL> DROP TABLE EMP
```

```
2
```

```
SQL> DROP TABLE EMP;
```

Table dropped.

```
SQL> DESC EMP
```

ERROR:

ORA-04043: object EMP does not exist

# Keys in Database

---

The different types of keys in DBMS are –

- Primary Key
- Candidate Key
- Composite Key
- Foreign Key
- Super Key

SID	Name	Marks	Dept	Course
1	A	65	CS	C1
2	B	75	EE	C1
3	A	65	CS	C2
4	B	37	EE	C3
5	c	89	IT	C2

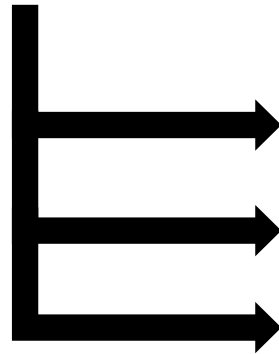
# Keys in Database

---

**Relation : Student**

**Why key is required?**

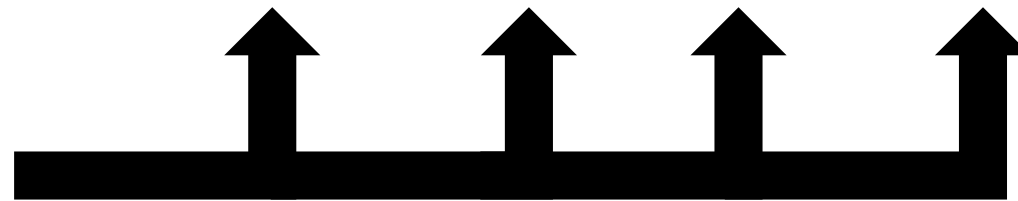
**Tuples/rows/records**



Name	Marks	Dept	Course
A	65	CS	C1
B	75	EE	C1
A	65	CS	C2
B	37	EE	C3
c	89	IT	C2

**To access data we need addressing.**

**Attributes/fields**



# Keys in Database

---

## Relation : Student

Name	Marks	Dept	Course
A	65	CS	C1
B	75	EE	C1
A	65	CS	C2
B	37	EE	C3
c	89	IT	C2

Want to access a record having name A

Want to access a record having Dept CS

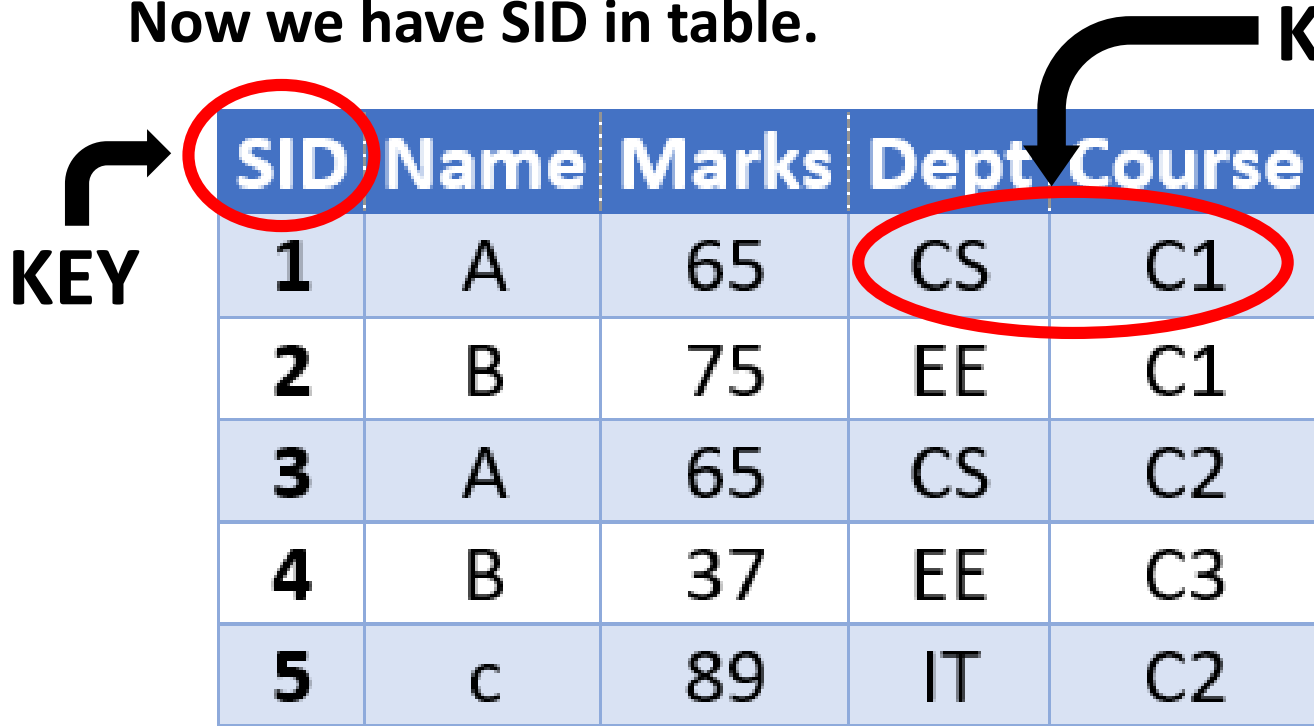
Create a problematic situation

Data will not be accessed.

To identify each record uniquely **KEY** is required.

# Keys in Database

Now we have SID in table.



SID	Name	Marks	Dept	Course
1	A	65	CS	C1
2	B	75	EE	C1
3	A	65	CS	C2
4	B	37	EE	C3
5	c	89	IT	C2

Consider, {Dept, Course}  
{CS,C1}

Each record can be identified uniquely

Consider, {Name, Marks}

Can we identify each record uniquely?

Consider, {Name, Marks, Dept, Course}

KEY

Using SID it is possible to access single record.

An attribute or set of attributes which uniquely identifies each record in the table or relation.



# Keys in Database

The different types of keys in DBMS are –

- Super Key -> key

SID – SK

{SID, Name} - SK

{SID, Marks} – SK

{Name, Marks, Dept} - ??

{Marks, Course, Dept} - ??

SID	Name	Marks	Dept	Course
1	A	65	CS	C1
2	B	75	EE	C1
3	A	65	CS	C2
4	B	37	EE	C3
5	c	89	IT	C2

How many maximum number of Super key is possible in relationship?  ${}^nC_r = \frac{n!}{r!(n-r)!}$

$${}^5C_1 + {}^5C_2 + {}^5C_3 + {}^5C_4 + {}^5C_5 = 5 + 10 + 10 + 5 + 1 = 31$$

How many maximum number of Super key is possible in relationship if attributes are 4?  **$2^n - 1$**

# Keys in Database

---

**R(A, B, C, D)**

**1 1 5 1**

**2 1 7 1**

**3 1 7 1**

**4 2 7 1**

**5 2 5 1**

**6 2 5 2**

Super keys in this relation are

**{A} {A,B} {A,C} {A,D}**

**{A,B,C} {A,C,D} {A,B,D}**

**{A,B,C,D} ~~{B,C,D}~~** Subset is also not a SK such as **{B,C} {B,D} {C,D}**

**Candidate key**

**It is a super key whose proper subset is not a super key**

**Also called as minimal key**

**R(A, B, C)**

**1 1 1**

**2 1 2**

**3 2 1**

**4 2 2**

# Keys in Database

---

R(A, B, C)	Super keys are		Super keys	Candidate key
1 1 1	{A, AB, AC,	ABC	yes	no
2 1 2	ABC, BC}	AC	yes	no
3 2 1		A	yes	yes
4 2 2		BC	yes	yes
S1 = {1,2,3}		AB	yes	no

S2 = {1,2}

$S2 \subseteq S1$

Proper subset  $S2 \subset S1$

$S2 \subseteq S1$

$S1 \not\subseteq S2$

Candidate Keys are A and BC

Candidate key is not minimum key but it is minimal key

Minimal means as minimum as possible

Every CK is a SK but not every SK is a CK

One CK that have no NULL value can be selected as PK

# Keys in Database

---

## Primary Key

**PRIMARY KEY** is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

### Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

StudID	Roll No	First Name	LastName
1	11	Tom	Price
2	12	Nick	Wright
3	13	Dana	Natan

# Keys in Database

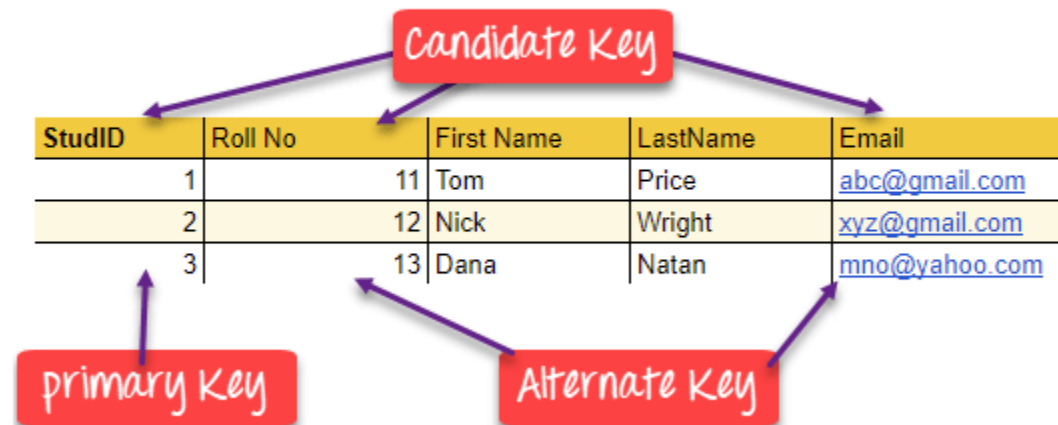
## Super Key

**Super Key** is defined as a set of attributes within a table that can uniquely identify each record within a table.

## Candidate Key

**Candidate keys** are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.

- A candidate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns(attributes).



# Keys in Database

---

## Foreign Key

A foreign key is an attribute value in a table that acts as the primary key in another table. Hence, the foreign key is useful in linking together two tables. Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.

- Foreign keys are the column of the table which is used to point to the primary key of another table.
- In a institute, every teacher works in a specific department, and teacher and department are two different entities. So we can't store the information of the department in the teacher table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the Department table, DeptCode as a new attribute in the Teacher table.

### PRIMARY KEY

DeptCode	DeptName
001	Science
002	English
005	Computer

### PRIMARY KEY

### FOREIGN KEY

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

# Keys in Database

---

## Composite Key

Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite key**. But the attributes which together form the **Composite key** are not a key independently or individually.

## Compound Key ??

Example:

**COMPOSITE KEY**

OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

# Keys in Database

---

## Primary Key

```
SQL> CREATE TABLE COURSES
  2  (CCODE VARCHAR2(10) PRIMARY KEY,
  3  CNAME VARCHAR2(50) NOT NULL,
  4  COURSEFEE NUMBER(6),
  5  DURATION NUMBER(3),
  6  PREREQ VARCHAR2(100));
```

Table created.

```
SQL> CREATE TABLE COURSES
  2  (CCODE VARCHAR2(10),
  3  CNAME VARCHAR2(50) NOT NULL,
  4  COURSEFEE NUMBER(6),
  5  DURATION NUMBER(3),
  6  PREREQ VARCHAR2(100),
  7  PRIMARY KEY (CCODE));
```

Table created.

```
SQL> CREATE TABLE COURSES
  2  (CCODE VARCHAR2(10),
  3  CNAME VARCHAR2(50) NOT NULL,
  4  COURSEFEE NUMBER(6),
  5  DURATION NUMBER(3),
  6  PREREQ VARCHAR2(100),
  7  CONSTRAINT PK_CODE PRIMARY KEY (CCODE));
```

Table created.

```
SQL> CREATE TABLE COURSES
  2  (CCODE VARCHAR2(10),
  3  CNAME VARCHAR2(50) NOT NULL,
  4  COURSEFEE NUMBER(6),
  5  DURATION NUMBER(3),
  6  PREREQ VARCHAR2(100),
  7  PRIMARY KEY (CCODE,CNAME));
```

Table created.

```
SQL> ALTER TABLE COURSES
  2  ADD PRIMARY KEY (CCODE);
```

Table altered.

```
SQL> ALTER TABLE COURSES
  2  DROP PRIMARY KEY;
```

Table altered.



# Keys in Database

---

## Foreign Key

```
SQL> CREATE TABLE BATCHES
2  (BATCHCODE VARCHAR2(15) CONSTRAINT BATCHES_PK PRIMARY KEY,
3  CCODE VARCHAR2(10) REFERENCES COURSES(CCODE));

Table created.
```

```
SQL> CREATE TABLE BATCHES
2  (BATCHCODE VARCHAR2(15) CONSTRAINT BATCHES_PK PRIMARY KEY,
3  CCODE VARCHAR2(10),
4  FOREIGN KEY (CCODE) REFERENCES COURSES(CCODE));

Table created.
```

```
SQL> CREATE TABLE BATCHES
2  (
3  BATCHCODE VARCHAR2(15) CONSTRAINT BATCHES_PK PRIMARY KEY,
4  CCODE VARCHAR2(10) CONSTRAINT BATCHES_CCODE_FK REFERENCES COURSES(CCODE)
5  );

Table created.
```

# Data Types in DBMS

---

DATA TYPE	DESCRIPTION
NUMBER	40 Digits with decimal and minus sign.
NUMBER(W)	Number of digits specified with decimal and minus sign.
NUMBER(W,D)	Number of digits, number of decimal digits specified with minus sign.
CHAR(W)	Fixed length character data. Maximum size is 2000 (bytes) and default is 1 (byte).
VARCHAR(W)	This variable is pre-reserved and maybe used by Oracle in later future, we should not use it now.
VARCHAR2(W)	String data of specified width. Maximum width is 4000 (bytes).
DATE	Date with default format 26-JAN-14.
LONG	String data of length 0-2 gigabytes. Only one LONG column per table. Can't be used with function, expression or WHERE clause.
RAW(W)	Binary data of specified length. Maximum width is 2000 (bytes).
LONG ROW	Binary data of length 0-2 gigabytes. Only one LONG ROW column per table.
CLOB	A character large object. Maximum size is 4 gigabytes.
BLOB	A binary large object. Maximum size is 4 gigabytes.
BFILE	Contains a locator to a large binary file stored outside the database. Maximum size is 4 gigabytes.

# SQL: Data Manipulation Language (DML)

---

- **Data Manipulation Language (DML)** commands are used **for accessing and manipulating the data stored in the database.**
- The DML commands are *not auto-committed* that means it can't permanently save all the changes in the database. They can be rollback.
- **Data Manipulation Language (DML)**
  - **SELECT** - to retrieve data from the database
  - **INSERT** - to insert a new row into a table
  - **UPDATE** - to update existing row in a table
  - **DELETE** - to delete a row from a table

← **Data Query language (DQL)**

**Modification of Database**

# SQL: Data Manipulation Language (DML)

---

- **Data Manipulation Language (DML)** commands are used **for accessing and manipulating the data stored in the database.**
  - The DML commands are *not auto-committed* that means it can't permanently save all the changes in the database. They can be rollback.
  - **Data Manipulation Language (DML)**
    - **SELECT** - to retrieve data from the database
    - **INSERT** - to insert a new row into a table
    - **UPDATE** - to update existing row in a table
    - **DELETE** - to delete a row from a table
- ← **Data Query language (DQL)**
- Modification of Database**

Programmers call these DML operations "CRUD".

**CRUD** stands for:

**Create, Read, Update, Delete**

**Performed by:**

**INSERT, SELECT, UPDATE, DELETE**



# Insert Statements

---

- Insert command is used to Insert data/record into the database table
- Inserting values for the specific columns in the table. (Always include the columns which are not null)

```
Insert Into <Table-Name> (Fieldname1, Fieldname2, Fieldname3,..) Values (value1,  
value2,value3,..);
```

Example:

```
SQL> insert into dept (deptno,dept_name)values (20,'HR');
```

If we want to insert values in all columns then no need to specify column values , but order of column values should be in sync with the column names.

Example:

```
SQL> insert into dept values (20,'HR');
```

# Insert Statements

---

```
SQL> DESC EMPLOYEE;
```

Name	Null?	Type
-----	-----	-----
EMPID		NUMBER(4)
NAME		VARCHAR2(20)
BIRTHDATE		DATE

```
SQL> INSERT INTO EMPLOYEE
  2  (EMPID,NAME,BIRTHDATE)
  3  VALUES (129,'VIRAT','29-SEP-02');

1 row created.
```

```
SQL> SELECT * FROM EMPLOYEE;
```

EMPID	NAME	BIRTHDATE
-----	-----	-----
230	SACHIN	
340	RAKESH	20-JUN-08
129	VIRAT	29-SEP-02

```
SQL> INSERT INTO EMPLOYEE
  2  VALUES(340,'RAKESH','20-JUN-08');

1 row created.
```

# Insert Statements

**INSERT ALL** statement is used to add multiple rows with a single **INSERT** statement. The rows can be inserted into one table or multiple tables using only one SQL command.

## Syntax

```
INSERT ALL
  INTO mytable (column1, column2, column_n) VALUES (expr1, expr2, expr_n)
  INTO mytable (column1, column2, column_n) VALUES (expr1, expr2, expr_n)
  INTO mytable (column1, column2, column_n) VALUES (expr1, expr2, expr_n)
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO EMPLOYEE (EMPID,NAME,BIRTHDATE) VALUES (104,'HARSH','14-NOV-77')
  3  INTO EMPLOYEE (EMPID,NAME,BIRTHDATE) VALUES (105,'VANSH','28-DEC-99')
  4  SELECT * FROM DUAL;
```

2 rows created.

```
SQL> SELECT * FROM EMPLOYEE;
```

EMPID	NAME	BIRTHDATE
230	SACHIN	
340	RAKESH	20-JUN-08
129	VIRAT	29-SEP-02
103	RAJ	29-APR-02
104	HARSH	14-NOV-77
105	VANSH	28-DEC-99

6 rows selected.

# 'Insert- As- Select' Statement

---

'Insert – As –Select' statement allows to insert into a table using the input from another table. Record from one table will be inserted in another table.

```
INSERT INTO table-name (column-names)
SELECT column-names
FROM table-name
WHERE condition
```

Syntax:

```
Insert into <new_table_name> (Select <columnnames> from <old_table_name>);
```

Example:

```
SQL> insert into CopyOfEmployee select * from employee where emp_id > 100
```



# 'Insert- As- Select' Statement

```
SQL> SELECT * FROM B1;
```

ROLLNO	NAME	MARKS
1981001	DHVANI	18
1981002	YAASMIN	19
1981003	SAKSHI	20
1981004	SAISH	12
1981005	NIDHI	18

```
SQL> SELECT * FROM B2;
```

ROLLNO	NAME	MARKS
1981031	VAISHNAVI	19
1981032	JHENIL	19
1981033	BHARGAVI	19
1981034	ISHITA	20
1981035	SAKSHI	18
1981036	MISHITA	19

```
SQL> INSERT INTO CSE  
2 SELECT * FROM B1;
```

5 rows created.

```
SQL>
```

```
SQL> CREATE TABLE CSE  
2 (ROLLNO NUMBER(8),  
3 NAME VARCHAR2(12),  
4
```

```
SQL>
```

```
SQL> SELECT * FROM CSE;
```

ROLLNO	NAME	MARKS
1981001	DHVANI	18
1981002	YAASMIN	19
1981003	SAKSHI	20
1981004	SAISH	12
1981005	NIDHI	18

# Update Statements

---

- Update statement updates/modify the existing data in the tables. Using these statements we can update the value of a single column or multiple columns in a single statement
- Updating single column

Syntax:

```
UPDATE <table name> Set <Field Name> = <Value> Where <Condition>;
```



**Note:**1. Without where clause all the rows will get updated

2. Updating multiple column [while updating more than one column, the column must be separated by comma operator]

```
Example: SQL> update dept set loc='Pink City', dept_name= 'HR' where deptno=20;
```

## Update Statements

---

**Note:** Always use the WHERE clause with the UPDATE statement to update the selected rows, otherwise all the rows would be affected.

### Syntax:

**UPDATE** *table\_name*

**SET** *column1 = value1, column2 = value2, ..... columnN = valueN*

**WHERE** *condition;*

# Update Statements

```
SQL> SELECT * FROM CSE;
```

ROLLNO	NAME	MARKS
1981001	DHVANI	18
1981002	YAASMIN	19
1981003	SAKSHI	20
1981004	SAISH	12
1981005	NIDHI	18

```
SQL> UPDATE CSE  
2 SET NAME = 'KULSUM';
```

5 rows updated.

```
SQL> SELECT * FROM CSE;
```

ROLLNO	NAME	MARKS
1981001	KULSUM	18
1981002	KULSUM	19
1981003	KULSUM	20
1981004	KULSUM	12
1981005	KULSUM	18

```
SQL> SELECT * FROM B2;
```

ROLLNO	NAME	MARKS
1981031	VAISHNAVI	19
1981032	JHENIL	19
1981033	BHARGAVI	19
1981034	ISHITA	20
1981035	SAKSHI	18
1981036	MISHITA	19

```
SQL> UPDATE B2  
2 SET NAME = 'SAKSHITA'  
3 WHERE ROLLNO = 1981035;
```

```
SQL> SELECT * FROM B2;
```

ROLLNO	NAME	MARKS
1981031	VAISHNAVI	19
1981032	JHENIL	19
1981033	BHARGAVI	19
1981034	ISHITA	20
1981035	SAKSHITA	18
1981036	MISHITA	19

```
SQL> UPDATE B2  
2 SET NAME = 'ZENIL', MARKS = 20  
3 WHERE ROLLNO = 1981031;
```

1 row updated.

```
SQL> SELECT * FROM B2;
```

ROLLNO	NAME	MARKS
1981031	ZENIL	20
1981032	JHENIL	19
1981033	BHARGAVI	19
1981034	ISHITA	20
1981035	SAKSHITA	18
1981036	MISHITA	19

# Delete Statements

Delete commands help to delete rows/record from database table. Delete Statements can be executed with or without where conditions. Execution of delete commands without where condition will remove all the records/rows from the table

## Syntax:

```
Delete from <table name> [where <condition>];
```

## Example:

a) to delete all rows (Deleting records without where condition):

```
SQL> delete from dept;
```

b) conditional deletion (deleting records with where condition):

```
SQL> delete from dept where loc='Pink City';
```

# Delete Statements

---

- The **DELETE** statement is used to delete existing records in a table.
- Note: *Always use the WHERE clause with a DELETE statement to delete the selected rows, otherwise all the records would be deleted.*

Syntax:

```
DELETE FROM table_name  
WHERE condition;
```

# Delete Statements

---

```
SQL> SELECT * FROM B1  
2 ;
```

ROLLNO	NAME	MARKS
1981001	DHVANI	18
1981002	YAASMIN	19
1981003	SAKSHI	20
1981004	SAISH	12
1981005	NIDHI	18

```
SQL> DELETE FROM B1  
2 WHERE NAME = 'NIDHI';
```

```
SQL> SELECT * FROM B1;
```

ROLLNO	NAME	MARKS
1981001	DHVANI	18
1981002	YAASMIN	19
1981003	SAKSHI	20
1981004	SAISH	12

```
SQL> DELETE FROM B1;
```

```
4 rows deleted.
```

```
SQL> SELECT * FROM B1;
```

```
no rows selected
```



# Select Statement

---

- ❑ **SELECT** statement is **used to select a set of data from a database table**. Or Simply **SELECT** statement is **used to retrieve data from a database table**.
  - ❑ It **returns** data in the form of a **result table**. These result tables are called **result-sets**.
- ❑ **SELECT** is also called **DQL** because it is used to **query** information from a database table
- ❑ **SELECT** statement specifies **column names**, and **FROM** specifies **table name**
- ❑ **SELECT** command is used with different **Conditions** and **CLAUSES**.

## **Basic Syntax:**

**To retrieve selected fields from the table:**

```
SELECT column1, column2, ... columnN  
FROM table_name(s);
```

**To retrieve all the fields from the table:**

```
SELECT *  
FROM table_name(s);
```



# Select Statement

## Example: SELECT

### Example 1:

```
SQL> SELECT * FROM B2;
```

ROLLNO	NAME	MARKS
1981031	ZENIL	20
1981032	JHENIL	19
1981033	BHARGAVI	19
1981034	ISHITA	20
1981035	SAKSHITA	18
1981036	MISHITA	19

6 rows selected.

### Example 2:

```
SQL> SELECT NAME, MARKS FROM B2;
```

NAME	MARKS
ZENIL	20
JHENIL	19
BHARGAVI	19
ISHITA	20
SAKSHITA	18
MISHITA	19

6 rows selected.

# Select Statement

---

Using Alias name for a field:

Syntax:

```
Select <col1> <alias name 1> , <col2> < alias name 2> from < tab1>;
```

```
SQL> SELECT NAME, MARKS AS FCN FROM B2;
```

NAME	FCN
ZENIL	20
JHENIL	19
BHARGAVI	19
ISHITA	20
SAKSHITA	18
MISHITA	19

# WHERE CLAUSE

- **WHERE Clause** is used to select a particular record based on a condition. It is used to filter records.
- **WHERE Clause** is used to specify a condition while fetching the data from a single table or by joining with multiple conditions
- The **WHERE** clause is not only used in the **SELECT** statement, but it is also used in the **UPDATE**, **DELETE** statement.
- Syntax: "SELECT with WHERE Clause"

**SELECT** column1, column2, ... columnN

**FROM** table\_name(s)

**WHERE** condition;

# Optional Clauses in **SELECT** statement

- ❑ [**WHERE Clause**] : It specifies which rows to retrieve by specifying **conditions**.
- ❑ [**GROUP BY Clause**] : **Groups** rows that share a **property** so that the **aggregate function** can be applied to each group.
- ❑ [**HAVING Clause**] : It **selects** among the **groups** defined by the **GROUP BY** clause by specifying **conditions**.
- ❑ [**ORDER BY**] : It specifies an order in which to return the rows.
- ❑ [**DISTINCT Clause**]: It is used to remove duplicates from results set of a **SELECT** statement. (**SELECT DISTINCT**)

# Exercise 1

---



Create following table as per given description and data.

**Table Name: Client\_book**

**Description: It is used to store client information**

Column Name	Data Type	Size	Attributes
client_no	Varchar2	5	Primary Key/First letter must start with 'C'
name	Varchar2	20	Not Null
address	Varchar2	20	
city	Varchar2	15	
pincode	number	6	
state	Varchar2	15	
bal_due	number	10,2	

# Exercise 1



## Data for table 'Client\_book'

Client_no	Name	City	Pincode	State	Balance_due
C0001	Yash Patel	Bhavnagar	364001	Gujarat	15000
C0002	Ravi Shulka	Indore	451010	Madhya Pradesh	0
C0003	Parvati Bhat	Kalyan	421004	Maharashtra	5000
C0004	Parshva Shah	Surat	394010	Gujarat	0
C0005	Kudrat Sethia	Kota	324006	Rajasthan	2000
C0006	Satendra Lohya	Gwalior	474008	Madhya Pradesh	0
C0007	Bablu Joshi	Lonavla	410405	Maharashtra	0
C0008	Jinal Dave	Vapi	396193	Gujarat	3000
C0009	Romil Monani	Valsad	396055	Gujarat	7000
C0010	Kiran Surve	Nashik	422002	Maharashtra	4000
C0011	Rohit Yadav	Vasai	401207	Maharashtra	0
C0012	Kailash Poonia	Bikaner	334005	Rajasthan	8000

# Exercise 1

---

Create following table as per given description and data.

Table Name: **product\_book**

Description: It is used to store product information

Column Name	Data Type	Size	Attributes
product_no	Varchar2	5	Primary Key/First letter must start with 'P'
name	Varchar2	20	Not Null
profit_percent	Varchar2	20	Not Null
quantity	Varchar2	15	Not Null
reorder_level	number	6	Not Null
sell_price	Varchar2	15	Not Null, can not be 0
cost_due	number	10,2	Not Null, cannot be 0

# Exercise 1

## Data for table 'Product\_book'

Product_no	Name	Profit_percent	Quantity	Reorder_level	Sell_price	Cost_price
P0001	CD Drive	5	100	7	900	800
P0002	Monitor	6	40	1	5500	5000
P0003	Mouse	5	150	9	300	250
P0004	Keyboard	5	75	6	400	300
P0005	External SSD	3	50	3	5200	5000
P0006	Cabinet	5	90	0	3500	3000
P0007	Keyboard	5	120	2	400	300
P0008	Mouse	4	70	8	350	250
P0009	Cabinet	7	40	2	4000	3000
P0010	Monitor	3	20	3	6000	5000



# Exercise 1



Create following table as per given description and data.

**Table Name:** saleman\_book

**Description:** It is used to store product information.

Column Name	Data Type	Size	Attributes
saleman_no	Varchar2	5	Primary Key/First letter must start with 'S'
s_name	Varchar2	20	Not Null
city	Varchar2	15	
state	Varchar2	15	
salary	number	6	Not Null, cannot be 0
target	number	5	Not Null, can not be 0
sales_done	Number	5	Not Null

# Exercise 1

---



Data for table 'saleman\_book'

saleman_no	s_name	city	state	salary	target	sales_done
S0001	Radhe	Surat	Gujarat	2000	100	70
S0002	Prem	Gwalior	Madhya Pradesh	5000	250	140
S0003	Karan	Kalyan	Maharashtra	3000	300	200
S0004	Ratan	Vapi	Gujarat	4000	50	40
S0005	Rahul	Kota	Rajasthan	6000	150	100
S0006	Sanju	Indore	Madhya Pradesh	3000	100	80