Explain DDL, DML, DCL and DQL. OR

Describe component of SQL.

- SQL stands for Structured Query Language.
- SQL is a standard language for accessing and manipulating databases.

DDL (Data Definition Language)

- It is a set of SQL commands used to create, modify and delete database objects such as tables, views, indices, etc.
- It is normally used by DBA and database designers.
- It provides commands like:

  CREATE: to create objects in a database.
  ALTER: to alter the schema, or logical structure, of the database.
  DROP: to delete objects from the database.
  TRUNCATE: to remove all records from the table.

DML (Data manipulation Language)

- It is a set of SQL commands used to insert, modify and delete data in a database.
- It is normally used by general users who are accessing database via pre-developed applications.
- It provides commands like:

  INSERT: to insert data into a table.
  UPDATE: to modify existing data in a table.
  DELETE: to delete records from a table.
  LOCK: to lock tables to provide concurrency control among multiple users.

DQL (Data Query Language)

- It is a component of SQL that allows data retrieval from the database.
- It provides command like **SELECT.** This command is a heart of SQL, and allows data retrieval in different ways.

DCL (Data Control Language)

- It is set of SQL commands used to control access to data and database. Occasionally DCL commands are grouped with DML commands.
- It provides commands like:

  COMMIT: to save work permanently.
  ROLLBACK: to undo work and restore database to previous state.
  SAVEPOINT: to identify a point in a transaction to which work can be undone.
  GRANT: to give access privileges to users on the database.
  REVOKE: to withdraw access privileges given to users on the database.

Describe different data types in SQL.

- Oracle supports a set of basic data types.
- There are four basic data type available in SQL
  - Numerical
  - Binary
  - Character
  - Date

Numerical Data Types:

Used to store zero, negative and positive numerical values. These values can be fixed-point (whole numbers) or floating-point (real numbers).

No	Data Type	Represent
1	NUMBER(P,S)	Floating-point number. P: precision, i.e. maximum number of digits in a number. Precision can be up to 38 digits. S: scale, i.e. number of digits to the right of the decimal point. Ex: number(6,2) = 1234.79
2	NUMBER(P)	Fixed-lenght number. Ex: number(6) = 123456
3	NUMBER	Floating-point number with a precision of 38 digits.

Character/String data types

No.	Data Type	**Description**
1	CHAR(size)	Stores character string of fixed length. Size represents the number of characters to be stored Default size is 1. ✓ Maximum length is 255 characters.
2	VARCHAR(size)/ VARCHAR2(size)	✓ Stores character string of variable length. ✓ More flexible than CHAR. ✓ No default size will be considered. So, size must be specified explicitly. ✓ Maximum length is 2000 characters.
3	LONG	✓ Stores large amount of character strings of variable length. ✓ Maximum length is up to 2 GB. ✓ Only one column per table can be defined as LONG.

Binary Data Types:

Examples of data which comes under binary type are images, audio, and video files.

No.	Data Type	**Description**
1	RAW	Stores binary type data. Maximum length is up to 255 bytes.
2	LONG RAW	Stores large amount of binary type data. Often referred as binary large object. Maximum length is up to 2 GB.

Date Data types

- ✓ Used to store date and time.
- ✓ The standard format is DD-MON-YY to store date, such as 1-JAN-11.
- ✓ The current date and time can be retrieved using function SYSDATE.
- ✓ Addition and subtraction operation are possible using number constants and other dates. For example, SYSDATE + 7 will add 7 to current date.

501	Akash S›odhoura	CEANS	9429/94513	Jamragar	10000	08/01/2012	Lecturer	502
S02	Chintan Kanani	CNK	9090908585	Rajkot	B000	06/01/2009	HOD	500
S03	Vishal I'/ak\‹ana	VKLI	8671010867	Jetpur	12500	08.*01/2012	Lectu e	502
50d	Payal Boda	PfrIB	d5d6784512	Rajkot		06.1 5/201d	Lectu e	502
505	Suresh	SSS	/878989852	Baroda	1122	01.*1 8/2009	Rao Assistant	500
500	Nilesh GambhaYa	NING	9925599255	Ra kot	22222	08.*01/2008	VePincpa	

➢ **CREATE TABLE QUERY:**

- ✓ This statement is used to create a new table.
- ✓ **Syntax:**

```
CREATE TABLE TABLENAME (COLUMNNAME1 DATATYPE (SIZE), COLUMNNAME2 DATATYPE (SIZE), .., COLUMNNAMEN DATATYPE (SIZE));
```

- ✓ **Example:**

```
CREATE TABLE STAFF_DETAILS (STAFF_ID NUMBER, STAFF_NAME VARCHAR2(20), STAFF_INITIAL VARCHAR2(5), STAFF_MNO INT(10), STAFF_ADDRESS VARCHAR2(30), STAFF_SALARY INT(6), STAFF_HIREDATE DATE,STAFF_TYPE VARCHAR2(10)) ;
```

➢ **DESCRIBE TABLE QUERY:**

- ✓ This statement is to used verify whether table has been created according to specification .
- ✓ **Syntax:**

```
DESCRIBE tableName;
```

- ✓ **Example:**

```
DESCRIBE STAFF_DETAILS;
```

- Inserting New Rows:

✓ This statement is used to insert new rows into created table as per specification .

✓ **Syntax:**

```
INSERT INTO TABLENAME (COLUMN1, COLUMN2,......, COLUMNN) VALUES
(EXPRESSION1,EXPRESSION2,. ... ,EXPRESSIONN);
```

✓ **Example:**

```
INSERT INTO STAFF_DETAILS
(STAFF_ID,STAFF_NAME,STAFF_INTIAL,STAFF_MNO,STAFF_ADDRESS)
VALUES (507,'HETASVI RIBADIA','CEHRR',12345678,'RAJKOT');
```

- Display All Rows and All Columns:

✓ This statement is used to retrieves all rows and all columns of given table

✓ **Syntax:**

```
SELECT * FROM TABLENAME;
```

✓ **Example:**

```
SELECT * FROM STAFF DETAILS;
```

- Selected Columns and All Rows:

✓ This statement retrieves only specified columns and all rows of given table .

✓ **Syntax:**

```
SELECT COLUMN1, COLUMN2,......,COLUMNN FROM  TABLENAME;
```

✓ **Example:**

```
SELECT STAFF_NAME,STAFF_INITIAL FROM  STAFF_DETAILS;
```

- Selected Rows and All Columns :

✓ This statement retrieves only specified rows and all columns of given table .

✓ **Syntax:**

```
SELECT * FROM  TABLENAME WHERE  CONDITION;
```

✓ **Example:**

```
SELECT * FROM STAFF DETAILS WHERE STAFF ID = 101;
```

- Selected Columns , Selected Rows:

✓ This statement retrieves only selected columns as specified with specific rows only in the given table

✓ **Syntax:**

```
SELECT COLUMN1,COLUMN2,......,COLUMN FROM  TABLENAME WHERE
CONDITION;
```

✓ **Example:**

```
SELECT STAFF_NAME FROM STAFF_DETAILS WHERE
STAFF_INITIAL='CEANS';
```

- Update All Rows:

✓ This statement updates all rows from given table.

✓ **Syntax:**

```
UPDATE  TABLENAME SET  COLUMN1 = VALUE1, COLUMN2 = VALUE2;
```

✓ **Example:**

```
UPDATE  STAFF DETAILS SET STAFF SALARY=10000;
```

- **Update Specific Rows:**

✓ This statement is used to update specific rows to satisfy condition.

✓ **Syntax:**

```
UPDATE TABLENAME SET COLUMN1 = VALUE1, COLUMN2 = VALUE2
WHERE CONDITION;
```

✓ **Example:**

```
UPDATE STAFF DETAILS SET STAFF ADDRESS='RAJKOT' WHERE
STAFF ID=105;
```

- **Deleting Specific Rows:**

✓ This statement is used to delete specific rows from table as per given condition.

✓ **Syntax:**

```
DELETE FROM TABLENAME WHERE CONDITION;
```

✓ **Example:**

```
DELETE FROM STAFF_DETAILS WHERE STAFF_ID = 105;
```

- Deleting All Rows:

✓ This statement is used to delete all rows from table.

✓ **Syntax:**

```
DELETE FROM TABLENAME;
```

✓ **Example:**

```
DELETE FROM STAFF_DETAILS;
```

- Truncating All Rows:

✓ This statement is used to remove all records from given table.

✓ Note: Records which are deleted using this command cannot be rollback.

✓ **Syntax:**

```
TRUNCATE TABLE TABLENAME;
```

✓ **Example:**

```
TRUNCATE TABLE STAFF_DETAILS;
```

- Destroying All Rows:

✓ This statement is used to destroy all records along with structure of table.

✓ **Syntax:**

```
DROPTABLE TABLENAME;
```

✓ **Example:**

```
DROPTABLE STAFF DETAIL;
```

- Adding New Columns:

✓ This statement is used to add new columns in any table.

✓ **Syntax:**

```
ALTER TABLE TABLENAME ADD (NEWCOLUMNNAME DATATYPE1 (SIZE),
NEWCOLUMNNAME DATATYPE2 (SIZE) ... );
```

✓ **Example:**

```
ALTER TABLE STAFF_DETAILS ADD (STAFF_TYPE VARCHAR2 (10));
```

- **Modifying Existing Columns:**

✓ This statement is used to set newDatatype and newsize as datatype and size for specified column respectively.

✓ **Syntax:**

```
ALTER TABLE TABLENAME MODIFY (COLUMNNAME NEWDATATYPE (NEWSIZE)
```

✓ **Example:**

```
ALTER TABLE STAFF_DETAILS MODIFY (STAFF_TYPE VARCHAR (20));
```

- **Dropping Columns:**

✓ This statement is used to delete an existing column from table along with data held by that column.

✓ **Syntax:**

```
ALTER TABLE TABLENAMEDROP COLUMN COLUMNNAME;
```

✓ **Example:**

```
ALTER TABLE STAFF_DETAILS DROP COLUMN STAFF_TYPE;
```

SQL Functions

- SQL has many built-in functions for performing calculations on data.

Numeric Functions :		
Name	**Description**	**Example**
Abs(n)	Returns the absolute value of n.	`SELECT ABS(—15) FROM DUAL;`
Power (m, n)	Returns m raised to n^{th} power.	`SELECT POWER(3,2) FROM DUAL;`
Round (n, m)	Returns n rounded to m places the right of decimal point.	SELECT ROUND (15 . 9 I , 1) FROM DB A,
Trunc(m, n)	m is truncated to n places to the right of a decimal point	SEIuECT FRANC (15 . 7 32, 2) FR OM DUAL ;
Sqrt(n)	Returns square root of n.	SELECT SQR7 (25) FROM DUAL;
Exp(n)	Returns e raised to the n^{th} power, e=2.17828183.	SELECT EXP (1) FROM DUAL ;
Mod(n, m)	Returns remainder of n divided by m.	`SELECT MOD(10,2) FROM DUAL;`
Ceil(n)	Returns the smallest integer value that is greater than or equal to a n.	SEIuECT CEI L (25. 2) FROM D B A,
Floor(n)	Returns the greatest integer value that is less than or equal to a n.	SEIuECT FLOOR (2 5 . 2) FROM D B A,

Conversion Function:		
Name	**Description**	**Example**
TO_CHAR	Converts numeric and date values to a character string value. It cannot be used for calculations since it is a string value.	SEIuECT STAFF NAME , T@CCHAR (S TAFF_HI REDATE , MM/ DD / YY77) HI REDATE FROM STAFF_DETAI US WHERE WHERE STARF_I D= ' 1 0 1 ;
TO_DATE	Converts a valid numeric and character values to date value.	SELECT TO DATE (2 0 03/ 0 7 / 0 9 , ’ BY/MM/DD ’) FROM DUAL;
TO_NUMBER	Converts a character string toanumberformat	SELECT TO_NUMBER(STAFF_SALARY,'999999') FROM STAFF_DETAILS;

Aggregate Function:		
Name	**Description**	**Example**
Max(column Name)	Returns maximum values for a given column.	SELECT MAX(STAFF_SALARY) FROM STAFF_DETAILS;
Min(column Name)	Returns minimum values for a given column.	SELECT MIN(STAFF_SALARY) FROM STAFF_DETAILS;
sum(column Name)	Returns sum of all values for a given column.	SELECT SUM(STAFF_SALARY) FROM STAFF_DETAILS;
avg(columnName)	Returns average of all values for a given column.	SELECT AVG(STAFF_SALARY) FROM STAFF_DETAILS;
count(*)	Returns number of rows in a table including duplicates and having null values.	SELECT COUNT(*) FROM STAFF_DETAILS;
count(column Name)	Returns number of rows where column does not contain null values.	SELECT COUNT(STAFF_SALARY) FROM STAFF_DETAILS;

Character Function		
Name	**Description**	**Example**
Length(str)	Returns the number of character in x.	SELECT LENGTH(’DIETDS’) FROM DUAL
Lower(str)	Converts the string to lower case.	SELECT LOWER(’DIETDS’) FROM DUAL;
Upper(str)	Converts the string to upper case.	SELECT UPPER(’DIETDS’) FROM DUAL;
Initcap (str)	Changes the first letter of a word in to capital.	SELECT INITCAP(’DIETDS’) FROM DUAL;
Substr(len,pos,str)	Returns the part of string	SELECT SUBSTR(’I LOVE C PROGRAMMING’,10,11) FROM DUAL;
Ltrim(set,str)	Remove all specified trim char from left side of the string.	SELECT LTRIM('DARSHAN DIPLOMA', 'DAR') FROM DUAL;

Rtrim(set,str)	Remove all specified trim char from right side of the string.	SELECT RTRIM (DARSHAN DIP / OMA ’¿ ’OMA) FROM DUAL;
Replace(to,from,str)	Looks for the str and replace the string every time it occurs.	SELECT REPLACE('DARSHAN DIPLOMA', 'DIPLOMA','COMPUTER') FROM DUAL;

Date Function:		
Name	**Description**	**Example**
MONTHS_BETWEEN	Finds the number of months betweem date1 and da date2. The result can be positive or negative.	SELECT MONTHS_BETWEEN('1—10—2017','1—10—2016') FROM DUAL; O/P : 12
ADD_MONTHS	Add ‘n’ numbers of calendar months to date	SELECT ADD_MONTHS('1—10—2017',5) FROM DUAL; O/P : 01/03/2018
NEXT_DAY	Find the date of the next specified day of the week	SELECT NEXT DAY('03—11—1990','SATURDAY') FROM DUAL; O/P 10—11—1990
LAST_DAY	Finds the date of the last day of the month that contains date	SELECT LAST_DAY('03—11—1990') FROM DUAL; O/P : 30/11/1990
ROUND	Function is used to get the date rounded to the unit specified by the format model.	SELECT ROUND (TO_DATE('03—11—1990'), 'MONTH') FROM DUAL; **01/11/1990** SELECT ROUND (TO_DATE('03—11—1990'),'YEAR') FROM DUAL; **O/P : 01/01/1990**

Explain SQL Operators.

- Operators are used inside an expression or condition to specify particular operations.
  1. Arithmetic Operators
  2. Relational Operators
  3. Logical Operators
  4. Range Searching Operators
  5. Set Searching Operators
  6. Character Operators

Arithmetic Operators

- Arithmetic operations can be performed while viewing table data, or while manipulating table data with insert, update or delete operation.

Operator	Specifies
+	Addition
	Subtraction
*	Multiplication
/	Division
( )	Enclosed operation

- **Example: Calculate new salary for each employee** after **receiving 10% rise and display it** along with employee id, name, and current salary.

```
SELECT STAFF_ID, STAFF_NAME, STAFF_SALARY "CURRENT SALARY",
(STAFF_SALARY + STAFF_SALARY * 0.1) "NEW SALARY" FROM
STAFF_DETAILS;
```

Relational Operators

- SQL supports following relational operators to perform comparisons among values.

Operator	Specifies
	Equals
!= or <>	Not equals
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

- These relational operators compare expressions and return one of three values: True, False or Unknown. An Unknown is returned when comparison is performed with a null value. Also, notice that both symbols, != and <>, represents "not equals".
- **Example:**

```
SELECT STAFF_NAME, STAFF_SALARY FROM STAFF_DETAILS WHERE
STAFF SALARY>=50000;
```

Logical Operators

- SQL provides three logical operators: AND, OR and NOT.

AND

- AND operator is used to combine two or more conditions in WHERE and HAVING clauses.
- AND requires all the conditions to be true to consider entire clause true.
- Example: Find the staff who are from Rajkot and Salary is greater than 10,000

```
SELECT STAFF NAME FROM STAFF WHERE STAFF SALARY>=10000 AND
STAFF ADDRESS='RAJKOT';
```

OR

- The OR operator is also used to combine two or more conditions in WHERE and HAVING clauses. OR requires anyone condition to be true to consider entire clause true.
- Example: Find the staff who are from Rajkot or Salary is greater than 10,000

```
SELECT STAFF_NAME FROM STAFF_DETAILS WHERE STAFF_SALARY>=10000
OR STAFF_ADDRESS='RAJKOT';
```

NOT

- The NOT operator is used to negate the result of any condition or group of conditions.
- Example: Find the list of whose joining is not there before 1-1-2011

```
SELECT STAFF NAME FROM STAFF DETAILS WHERE NOT
(STAFF_HIREDATE>='1—1—2012');
```

Range Searching Operator (BETWEEN Operator)

- This operator is used to select data that belong to some particular range.
- Syntax: ... co umnName **BETWEEN** a new Smt L AND upper Smt L;
- Selects rows that contain values within a specified lower and upper limit.
- The lower and upper limits are inclusive in range.
- Example: Find the list of staff whose joining is between 2010 to 2012

```
        sTAFF NA        STAFF DE  2LS WHERE STAFF HIREDATE
B:::E:N        ME 'RNM
```

Set Searching Operator (IN Operator)

- This operator is used to select data that belongs to some particular set of values.
- **Syntax:**

```
          columnName IN (valuel, value2,      , valueN )
```

- Selects rows that contain any value given in a set.
- This is similar to '='. But, '=' compares single value to another single value, while IN compares single value to a list (set) of values provided with IN predicate.
- Example: List out staff who are from Rajkot, Jamnagar and Junagadh.

```
SELECT * FROM STAFF_DETAILS WHERE STAFF_ADDRESS IN
('AHMEDABAD', 'VADODARA', 'SURAT' );
```

Character Operators

LIKE Operator (Pattern Matching Operator)

- Syntax: . c OI umnNare IuIKE P ate em ;
- Selects rows that contain values (strings) similar to a given pattern.
- This is similar to '='. But, the '=' operator compares for exact matching. While LIKE compares for pattern similarity.
- Pattern can be formed by using two wild card characters:

1. % (modulo) allows matching with any string having any number of characters, including zero.
2. (underscore) allows matching with any single character. The examples given below clarify the use of LIKE operator.

- **Example: Display employees having city name starting with 'A'.**
  ```
  SELECT STAFF NAME, STAFF ADDRESS FROM STAFF DETAILS WHERE
  STAFF ADDRESS LIKE ' a%';
  ```

Operator (Concatenation Operator)

- Syntax:
  ```
  STRING1    STRING2
  ```
- Concatenates, i.e. combines two strings.- Useful while displaying output.
- Strings can be constant as well as column names having character data type.
- **Example: Combines staff name and staff initial.**
  ```
  SELECT STAFF_NAME || '—' || STAFF_INITIAL "STAFF NAME WITH
  INITIAL" FROM STAFF_DETAILS;
  ```

Miscellaneous Function:		
Name	**Description**	**Example**
DECODE	Function decodes an expression in a way similar to the IF-THEN-ELSE logic used. It decodes expression after comparing it to each search value.	`SELECT STAFF_NAME,STAFF_SALARY, DECODE (STAFF_INITIAL,'CEANS',2*STAFF SALARY, 'CENJR',10*STAFF SALARY,STAFF_SALARY) REVISED_SALARY FROM STAFF_DETAILS;`
GREATEST	Returns the greatest values from the given expression	`SELECT GREATEST(10,20,50,20,30) FROM DUAL;`
LEAST	Returns the smallest values from the given expression	`SELECT LEAST(10,20,50,20,30) FROM DUAL;`
NVL	To convert a null value to an actual value	`SELECT STAFF_NAME,NVL(STAFF_SALARY,0) FROM STAFF_DETAILS;`
NVL2	Function examines the first expression. If the first expression is not null, then NVL2 function returns the second expression. If the first expression is null, then the third expression is returned. NVL2 (exp1, exp2, exp3)	`SELECT STAFF_NAME,STAFF_SALARY, NVL2(STAFF_SALARY,STAFF_SALARY,100*100) NVL2 FROM STAFF_DETAILS;`
NULLIF	Function compares two expressions. If they are equal, the function returns null. If they are not equal, the function returns the first expression.	`SELECT LENGTH(STAFF_SALARY) E1,LENGTH(STAFF_INITIAL) E2 NULLIF(LENGTH(STAFF_SALARY) ,LENGTH(STAFF_INITIAL)) AS "NULLIF EXAMPLE" FROM STAFF DETAILS;`
USER	Returns the username of the current user logged on.	`SELECT USER FROM DUAL;`

Explain Group By, Having and Order by With Example.

GROUP BY

- Syntax :

```
SELECT COLUMN1, COLUMN2,:., AGGREGATE FUNCTION(ARGUMENT)
FROM TABLENAME
GROUP BY COLUMN1, COLUMN2, :, COLUMNN;
```

- **Description:**

  The GROUP BY clause groups' records based distinct values for specified columns.

  In Other words, it creates a dataset — containing several sets of records grouped together based on a condition.

- Example

  Display staff count from different city:

```
SELECT STAFF_ADDRESS,
COUNT (STAFF_ADDRESS) FROM STAFF_DETAILS
GROUP BY STAFF ADDRESS
```

```
Jamnagar
Baroda          1
Jetpur
Rajkot          3
```

HAVING

- **Syntax:**

```
SELECT COLUMN1, COLUMN2,:, AGGREGATE FUNCTION (ARGUMENT)
FROM TABLENAME
GROUP BY COLUMN1, COLUMN2,      , COLUMN
HAVING CONDITION;
```

- **Description:**

  The HAVING clause filters groups based on the specified condition. (The WHERE clause filters rows (records) based on the specified condition in that clause).

  Each column specified in the HAVING clause must appear in the list of columns specified in the GROUP BY clause, or, it must appear within an aggregate function.

- Example
- **Displays the no. of staff who are from Rajkot**

```
SELECT STAFF_ADDRESS, COUNT (STAFF_ADDRESS)
FROM STAFF DETAILS
GROUP BY STAFF_ADDRESS
HAVING STAFF ADDRESS = 'Rajkot';
```

```
RakDt        3
```

ORDER BY

- Data can be viewed in a sorted order. Rows can be retrieved in ascending order or in descending order.
- **Syntax:**

```
SELECT * FROM TABLENAME ORDER BY COLUMN1 [ORDER], COLUMN2
[ORDER), ... , COLUMNN [ORDER];
```

- This statement retrieves data in a sorted manner.
- Rows are sorted based on values of columns specified with ORDER BY clause. By default, order is considered an Ascending order. To sort data in descending order, it is necessary to specify DESC as an order.

- When multiple columns are provided and some rows contain same data for column, then sorting is performed based on data of next column specified.
- **Example: Display all data on the basis of staff name in alphabetical order.**

```
SELECT    FROM STAFF_DETAILS ORDER BY  STAFF_NAME;
```

501	Akash Siddhpura	CEANS	9429794513	Jamnagar	10000	08'01.'2012	Lecturer	502
502	Chintan Kanani	CNK	9090908585	Rajkot	8000	06."01."2009	HOD	500
500	Nilesh Gambhava	NI'/1G	9925599255	Rajkot	22222	08"01 '2008	\/ce Principal	
504	Payal Boda	PI'1B	4546784512	Rajkol		06'15,'2014	Lecturer	502
505	Suresh	SSS	7878989852	Baroda	1122	0\.'13.'2009	Lab Assistant	500
503	Vishal I',1aI«vana	VKI 'I	8671010867	Jetpur	12500	08."01."2012	Lecturer	502

SQL Syntax

Eliminating Duplication	`SELECT DISTINCT COLUMN1,COLUMN2,... ,COLUMNN FROM TABLENAME;`	The DISTINCT word in statement removes duplicate rows from given table	`SELECT DISTINCT STAFF_ADDRESS FROM STAFF_DETAILS;`
Creating a Table from Another Table.	`CREATE TABLE NEWTABLENAME (COLUMN1,COLUMN2,... ,COLUMNN) AS SELECT COLUMN1,COLUMN2, ,COLUMNN FROM SOURCE TABLENAME WHERE CONDITION;`	This statement is used to create new table by newTableName from the given table specified by sourceTableName.	`CREATE TABLE STAFF_DETAIL AS SELECT * FROM STAFF_DETAILS;`
Inserting Data into a table from Another Table.	`INSERT INTO DESTINATIONTABLE (COLUMN1,COLUMN2,... ,COLUMNN) SELECT COLUMN1,COLUMN2, ,COLUMNN FROM SOURCETABLE WHERE CONDITION;`	This statement is used to copy records from source table to destination table	`INSERT INTO STAFF_DETAIL SELECT * FROM STAFF_DETAILS;`
Renaming a Table	`RENAME TABLENAME TO NEWTABLENAME;`	This statement is used to rename the table name to new table name	`RENAME STAFF_DETAILS TO STAFF2;`