# DELHI TECHNOLOGICAL UNIVERSITY


# DATABASE MANAGEMENT SYSTEM

# PRACTICAL FILE

**Submitted to:**

Dr. Shailender Kumar
Associate Professor

**Submitted by:**

Kunal Sinha
2K17/CO/A3/164

# INDEX

| S.No | Experiment | Done On | Checked on | Signature |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# PROGRAM 1

**AIM:**

1. Creation of a database and writing SQL Queries to retrieve information from the Database.
2. To implement Data Definition Language (DDL)
    - **a.** Create, Alter, Drop, Truncate.
    - **b.** To implement constraints
        - **i.** Primary Key
        - **ii.** Check
        - **iii.** Unique
        - **iv.** Not Null
        - **v.** Default

**THEORY:**

**A. CREATE TABLE:** The Create Table statement is used to create a new table in database.

**SYNTAX:**

```
CREATE table table_name
{COLUMN1 DATATYPE,
COLUMN2 DATATYPE
};
```

**B. ALTER TABLE:** The ALTER table statement is used to add ,delete or modify columns in an existing table.
It is also used to add and drop various constraints on existing table.

**SYNTAX:**

```
ALTER TABLE table_name
ADD column_name datatype;
```

**C. DROP TABLE:** The DROP table command is used to remove a table definition and all data ,indexes ,triggers, constraints and permission specifications for that table.

**SYNTAX:**

```
DROP TABLE table_name;
```

**D. TRUNCATE:** This command is used to delete complete data from an existing table

**SYNTAX:**

```
TRUNCATE TABLE
table_name;
```

**E. COLUMN CONSTRAINTS:** SQL constraints are used to specify therules for data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL.

- **PRIMARY KEY:** A combination of NOT NULL and UNIQUE. Uniquely defines each row in a tuple.
- **CHECK:** Ensures that all values in a column satisfies a specific condition.
- **UNIQUE:** Ensures that all values in a column are different.
- **NOTNULL:** Ensures that a column cannot have a NULL value
- **DEFAULT:** sets a default value for a column when no value is specified.

## a) DEPARTMENT

```
CREATE TABLE DEPARTMENT
(
dept_id INT PRIMARY KEY,
dept_name VARCHAR2(30) NOT NULL,
hod_id INT
);
```

Results Explain Describe Saved SQL History

**Table Created.**

(

## b) DOCTOR

```
CREATE TABLE DOCTOR
(
doc_id INT PRIMARY KEY,
doc_name VARCHAR(20) NOT NULL,
dept_id INT,
specialization VARCHAR(30),
designation VARCHAR(30),
qualification VARCHAR(30),
contact_no VARCHAR(13),
address VARCHAR(30)
);
```

Results Explain Describe Saved SQL History

0.09 seconds

## c) NURSE

```
CREATE TABLE NURSE
(
nurse_id INT PRIMARY KEY,
nurse_name VARCHAR(30) NOT NULL,
dept_id INT,
address VARCHAR(30),
dob DATE,
FOREIGN KEY(dept_id) REFERENCES DEPARTMENT(dept_id)
);
```

Results Explain Describe Saved SQL History

.49 seconds

## d) TEST

```
CREATE TABLE TEST
(
test_id INT PRIMARY KEY,
test_no VARCHAR2(20),
test_date DATE,
Report_date DATE
);
```

Results Explain Describe Saved SQL History

Table Created.

### e) WARD

```
CREATE TABLE WARD
(
ward_no INT PRIMARY KEY,
capacity INT,
ward_name VARCHAR(30),
ward_head VARCHAR(30),
employees INT
);
```

Results Explain Describe Saved SQL History

```
Table created.
```

1.44 seconds

- **Alter Table Command**

```
ALTER TABLE DOCTOR ADD doc_gender VARCHAR(8);
```

Results Explain Describe Saved SQL History

Table altered.

- **Truncate Table Command**

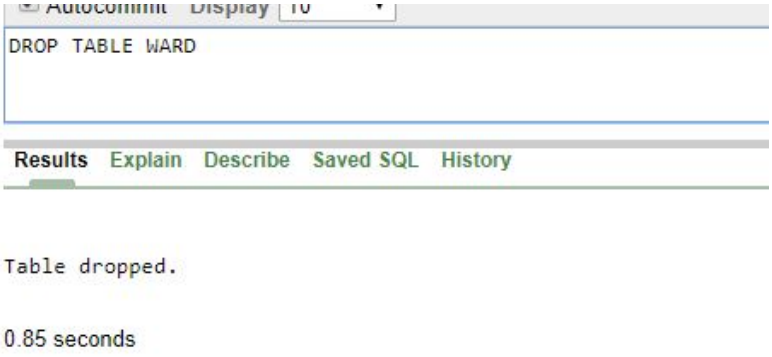
```
TRUNCATE TABLE WARD
```

Results Explain Describe Saved SQL History

```
Table truncated.
```

0.03 seconds

- **Drop Table Command**




**FINDINGS/LEARNING:**

a. The **CREATE** table defines the Schema of the table **DOCTOR, DEPARTMENT, NURSE, WARD** and **TEST**.
b. The **ALTER TABLE** command adds another column to the DOCTOR table.
c. **DROP TABLE** command deletes the schema of the table WARD.
d. **TRUNCATE** command deletes all the entities in the table WARD

**DISCUSSION:**

We get to run DDL queries for creation of database, alter, truncate the records and drop to physically remove the schema from the database.

**CONCLUSION:**

The Hospital Management system is created and DDL queries are successfully executed on it.

# PROGRAM 2

**AIM:** To implement Data Manipulation Language (DML)

- **c. INSERT**
- **d. SELECT**
- **e. DELETE**
- **f. UPDATE**.

**THEORY:**

1) **INSERT:** The Insert statement is used to insert data into table.

```
SYNTAX:
        INSERT INTO table_name
        VALUES (value1, value2,
value3, …..);
```

2) **SELECT:** The Select statement is used to retrieve data from table

```
SYNTAX:
        SELECT column_name(s)
        FROM table_name;
```

3) **DELETE:** The Delete statement is used to delete records from database table.

```
SYNTAX:
        DELETE from table_name
        WHERE some_column is
some_value;
```

4) **UPDATE:** The Update statement is used to update data in an existing table.

```
SYNTAX:
        UPDATE table_name
        SET  column1=value1, column
2=value2,…
        WHERE some_column is
some_value;
```

- **INSERT**

```
INSERT INTO DEPARTMENT (DEPT_ID, DEPT_NAME, HOD_ID) VALUES (6, 'DENTIST', 4)
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.
```

0.06 seconds

```
INSERT INTO WARD VALUES(205, 50, 'V', 'N', 12);
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.
```

0.00 seconds

```
INSERT INTO TEST VALUES(505, 20, TO_DATE('2018/01/08', 'YYYY/MM/DD'), TO_DATE('2018/01/20', 'YYYY/MM/DD'));
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.
```

0.02 seconds

```
INSERT INTO DOCTOR VALUES (4, 'NAMBIAR', 6, 'DENTIST', 'JUNIOR DOCTOR', 791264, null, 'FEMALE', 100000);
```

Results Explain Describe Saved SQL History

```
1 row(s) inserted.
```

0.81 seconds

- **SELECT**

```
SELECT * FROM DOCTOR;
```

Results Explain Describe Saved SQL History

| DOC_ID | DOC_NAME | DEPT_ID | SPECIALIZATION | DESIGNATION | CONTACT_NO | ADDRESS | DOC_GENDER | SALARY |
|---|---|---|---|---|---|---|---|---|
| 1 | MOHAN | 4 | CARDIOLOGIST | SENIOR DOCTOR | 9958627183 | Sector-70, Noida, NCR | MALE | 1000000 |
| 2 | BAID | 3 | CHIROPRACTOR | ASSISTANT DOCTOR | 778265 | - | MALE | 500000 |
| 3 | SRIVASTAVA | 2 | DERMATOLOGIST | ASSISTANT DOCTOR | 778247 | - | FEMALE | 150 |
| 4 | NAMBIAR | 6 | DENTIST | JUNIOR DOCTOR | 791264 | - | FEMALE | 100000 |

4 rows returned in 0.00 seconds CSV Export

```
SELECT * FROM TEST;
```

Results Explain Describe Saved SQL History

| TEST_ID | TEST_NO | TEST_DATE | REPORT_DATE |
|---|---|---|---|
| 501 | 11 | 07-JAN-18 | 15-JAN-18 |
| 502 | 13 | 07-JAN-18 | 15-JAN-18 |
| 503 | 12 | 07-JAN-18 | 15-JAN-18 |
| 504 | 16 | 07-JAN-18 | 15-JAN-18 |
| 505 | 20 | 08-JAN-18 | 20-JAN-18 |

5 rows returned in 0.00 seconds CSV Export

```
SELECT * FROM WARD;
```

Results Explain Describe Saved SQL History

| WARD_NO | CAPACITY | WARD_NAME | WARD_HEAD | EMPLOYEES |
|---|---|---|---|---|
| 201 | 50 | Z | J | 2 |
| 202 | 53 | Y | K | 12 |
| 203 | 54 | X | L | 3 |
| 204 | 50 | W | M | 10 |
| 205 | 50 | V | N | 12 |

5 rows returned in 0.13 seconds CSV Export

```
SELECT * FROM DEPARTMENT;
```

Results | Explain | Describe | Saved SQL | History

| DEPT_ID | DEPT_NAME | HOD_ID |
|---|---|---|
| 1 | NEUROLOGY | 4 |
| 2 | DERMATOLOGY | 2 |
| 3 | CHIROPRACTOR | 6 |
| 4 | CARDIOLOGY | 8 |
| 5 | GYNAECOLOGY | 2 |
| 6 | DENTIST | 4 |

6 rows returned in 0.01 seconds CSV Export

```
SELECT * FROM TEST;
```

Results | Explain | Describe | Saved SQL | History

| TEST_ID | TEST_NO | TEST_DATE | REPORT_DATE |
|---|---|---|---|
| 501 | 11 | 07-JAN-18 | 15-JAN-18 |
| 502 | 13 | 07-JAN-18 | 15-JAN-18 |
| 504 | 16 | 07-JAN-18 | 15-JAN-18 |
| 505 | 20 | 08-JAN-18 | 20-JAN-18 |

4 rows returned in 0.00 seconds CSV Export

```
SELECT * FROM NURSE;
```

Results | Explain | Describe | Saved SQL | History

| NURSE_ID | NURSE_NAME | DEPT_ID | ADDRESS | DOB | SALARY |
|---|---|---|---|---|---|
| 101 | A | 2 | BAWANA | - | - |
| 102 | B | 4 | BAWANA | - | - |
| 103 | C | 3 | BAWANA | - | - |
| 104 | D | 1 | BAWANA | - | - |
| 105 | E | 6 | BAWANA | - | - |

5 rows returned in 0.02 seconds CSV Export

- **UPDATE**

```
UPDATE DOCTOR SET doc_gender = 'FEMALE' WHERE doc_name = 'SRIVASTAVA';
```

Results Explain Describe Saved SQL History

```
1 row(s) updated.

0.21 seconds
```

- **DELETE**

```
DELETE FROM TEST WHERE TEST_ID = 503;
```

Results Explain Describe Saved SQL Histor

```
1 row(s) deleted.

0.00 seconds
```

**FINDINGS/LEARNING:**

a. The **INSERT** command is used to insert values in **DOCTOR, DEPARTMENT, NURSE, WARD** and **TEST**.
b. The **SELECT** command retrieves data from tables DOCTOR, DEPARTMENT, NURSE, WARD and TEST.
c. The **DELETE** command deletes the records from **TEST** where **TEST_ID=503**.
d. The **UPDATE** command updates the **DOCTOR** table and sets **DOC_GENDER='FEMALE'** where **DOC_NAME='SRIVASTAVA'**.

**DISCUSSION:**

The DML queries facilitate manipulations on the created database by inserting, selecting, deleting and updating the entries in various schemas of the Database .

**CONCLUSION:**

The DML queries are successfully executed on The Hospital Management system.

# PROGRAM 3

**AIM:** To implement Aggregate functions in SQL

- AVG( )
- COUNT( )
- MIN( )
- MAX( )
- SUM( )

**THEORY:**

SQL Aggregate functions return a single value calculated from the values in column.

| Function | Description |
|---|---|
| AVG( ) | Returns the average value |
| COUNT( ) | Returns the number of rows |
| MAX( ) | Returns the largest value |
| MIN( ) | Returns the smallest value |
| SUM( ) | Returns the sum |

- **AVERAGE**

```
SELECT AVG(EMPLOYEES) FROM WARD;
```

Results Explain Describe Saved SQL History

| AVG(EMPLOYEES) |
|---|
| 7.8 |

1 rows returned in 0.00 seconds CSV Export

- **COUNT**

```
SELECT COUNT(ADDRESS) FROM NURSE WHERE ADDRESS = 'BAWANA';
```

Results Explain Describe Saved SQL History

| COUNT(ADDRESS) |
| --- |
| 5 |

1 rows returned in 0.00 seconds CSV Export

- **SUM**

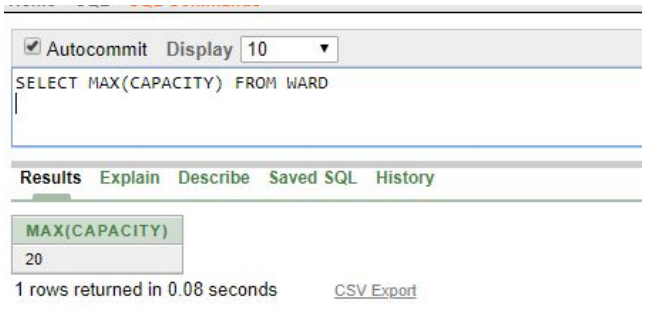
```
SELECT SUM(EMPLOYEES) FROM WARD;
```

Results Explain Describe Saved SQL History

| SUM(EMPLOYEES) |
| --- |
| 39 |

1 rows returned in 0.00 seconds CSV Export

- **MAX**



- **MIN**


**FINDINGS/LEARNING:**

- The Aggregate functions MAX( ), MIN( ), SUM( ), AVG( ) and COUNT( ) are used to perform calculations on the tables **DOCTOR, DEPARTMENT, NURSE, WARD** and **TEST**.

**CONCLUSION:**

The Aggregate functions are successfully executed on The Hospital Management system.

# PROGRAM 4

**AIM:** To implement the following functions in SQL

- String Functions
- Date Functions

**THEORY:**

**String Functions:** SQL string functions return a single value calculated from the values in column.

| Function | Description |
|---|---|
| LEN( )/LENGTH() | Returns the length of value in the text field |
| LOWER( )/LCASE() | Converts character data to lower case |
| SUBSTRING( ) | Extract characters from a text field |
| CONCAT( ) | Add two or more strings together |
| UPPER( )/UCASE() | Converts character data to upper case |

**Date Functions :** SQL date functions are used to find current date and time from system

- **LENGTH**

```
SELECT LENGTH(DOC_NAME) FROM DOCTOR;
```

Results Explain Describe Saved SQL History

| LENGTH(DOC_NAME) |
|---|
| 5 |
| 4 |
| 10 |
| 7 |

4 rows returned in 0.02 seconds CSV Export

- **CONCAT**

```
SELECT CONCAT(TEST_DATE,CONCAT(', ', REPORT_DATE)) AS "TEST AND REPORT DATES" FROM TEST;
```

Results Explain Describe Saved SQL History

| TEST AND REPORT DATES |
|---|
| 07-JAN-18, 15-JAN-18 |
| 07-JAN-18, 15-JAN-18 |
| 07-JAN-18, 15-JAN-18 |
| 08-JAN-18, 20-JAN-18 |

4 rows returned in 0.00 seconds CSV Export

- **LOWER**

```
SELECT LOWER(DEPT_NAME) AS "DEPARTMENT NAMES" FROM DEPARTMENT;
```

Results Explain Describe Saved SQL History

| DEPARTMENT NAMES |
|---|
| neurology |
| dermatology |
| chiropractor |
| cardiology |
| gynaecology |
| dentist |

6 rows returned in 0.00 seconds CSV Export

- **SUBSTRING**

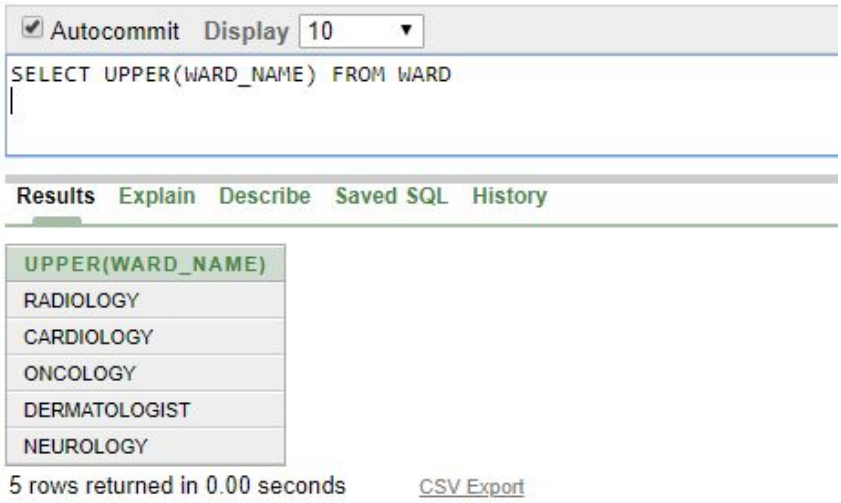
```
SELECT SUBSTR(SPECIALIZATION, 0, 4) AS "QUALI" FROM DOCTOR
```

Results Explain Describe Saved SQL History

| QUALI |
|---|
| CARD |
| CHIR |
| DERM |
| DENT |

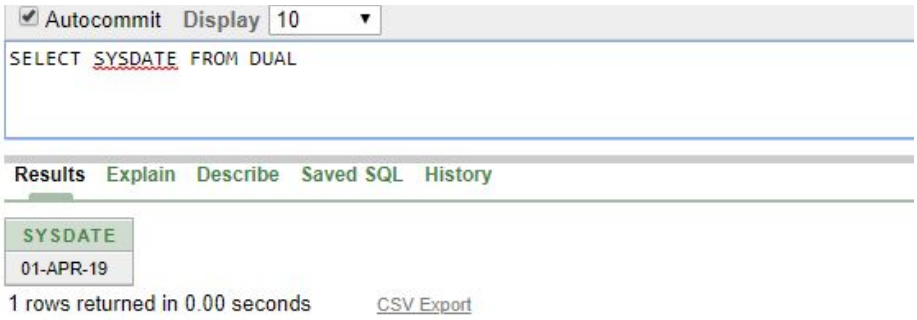
4 rows returned in 0.02 seconds CSV Export

- **UPPER**

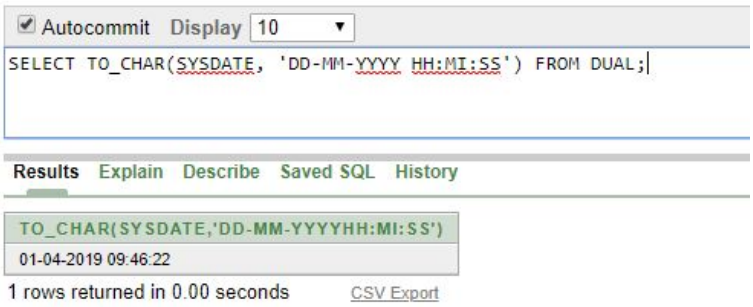



## DATE FUNCTIONS

- **SYSTEM DATE**




- **SYSTEM DATE AND TIME**



**FINDINGS/LEARNING:**

- String functions like **CONCAT**(), **UPPER**(), **LOWER**(), **SUBSTRING**(),**LENGTH**() are used to perform string operations on tables DEPARTMENT, WARD, DOCTOR, TEST.

- Date functions are used to find out current date and time from system.

**CONCLUSION:**

String and date functions are successfully executed on Hospital Management System

# PROGRAM 5

**AIM:**
To implement the following SQL joins on given relations

- Inner Join
- Left Outer Join
- Right Outer Join
- Full Outer Join
- Self Join

**THEORY:**

A. **INNER JOIN:** SQL INNER JOINS return all rows from multiple tables where the join condition is met

```
SYNTAX
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

B. **LEFT OUTER JOIN:** This type of join returns all rows from the LEFT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal (join condition is met)

```
SYNTAX
SELECT columns
FROM table1
LEFT OUTER JOIN table2
ON table1.column = table2.column;
```

C. **RIGHT OUTER JOIN:** This type of join returns all rows from the RIGHT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal (join condition is met).

```
SYNTAX
SELECT columns
FROM table1
RIGHT OUTER JOIN table2
ON table1.column = table2.column;
```

D. **FULL OUTER JOIN:** This type of join returns all rows from the LEFT-hand table and RIGHT-hand table with NULL values in place where the join condition is not met.

```
SYNTAX
SELECT columns
FROM table1
FULL OUTER JOIN table2
ON table1.column = table2.column;
```

E. **SELF JOIN:** A self JOIN is a regular join, but the table is joined with itself.

```
SYNTAX
    SELECT column_name(s)
    FROM table1 a, table1 b
    WHERE condition;
```

## DATABASE TABLES:

1. Customers

| customer_id | last_name | first_name | favorite_website |
|---|---|---|---|
| 4000 | Jackson | Joe | techonthenet.com |
| 5000 | Smith | Jane | digminecraft.com |
| 6000 | Ferguson | Samantha | bigactivities.com |
| 7000 | Reynolds | Allen | checkyourmath.com |
| 8000 | Anderson | Paige | NULL |
| 9000 | Johnson | Derek | techonthenet.com |

2. Orders

| order_id | customer_id | order_date |
|---|---|---|
| 1 | 7000 | 2016/04/18 |
| 2 | 5000 | 2016/04/18 |
| 3 | 8000 | 2016/04/19 |
| 4 | 4000 | 2016/04/20 |
| 5 | NULL | 2016/05/01 |

## QUERIES:

## Right Outer Join

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
RIGHT OUTER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

Execute SQL

**Query Results**

| customer_id | order_id | order_date |
|---|---|---|
| NULL | 5 | 2016/05/01 |
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |

## Left Outer Join

**SQL Statement**

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
LEFT OUTER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

Execute SQL

### Query Results

| customer_id | order_id | order_date |
|---|---|---|
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 6000 | NULL | NULL |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |
| 9000 | NULL | NULL |

## Inner Join

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
INNER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

Execute SQL

### Query Results

| customer_id | order_id | order_date |
|---|---|---|
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |

## Full Outer Join

```
SELECT customers.customer_id, orders.order_id, orders.order_date
FROM customers
FULL OUTER JOIN orders
ON customers.customer_id = orders.customer_id
ORDER BY customers.customer_id;
```

Execute SQL

**Query Results**

| customer_id | order_id | order_date |
|---|---|---|
| NULL | 5 | 2016/05/01 |
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 6000 | NULL | NULL |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |
| 9000 | NULL | NULL |

## Self Join

```
SELECT A.first_name AS CustomerName1, B.first_name AS CustomerName2, A.favorite_website
FROM customers A, customers B
WHERE A.customer_id <> B.customer_id
AND A.favorite_website = B.favorite_website;
```

Execute SQL

**Query Results**

| CustomerName1 | CustomerName2 | favorite_website |
|---|---|---|
| Joe | Derek | techonthenet.com |
| Derek | Joe | techonthenet.com |

**FINDINGS/LEARNING:**

a. The **INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN and SELF JOIN** command is used to join values in **CUSTOMERS** and **ORDERS**.

**DISCUSSION:**

SQL JOINS are used to retrieve data from multiple tables. A SQL JOIN is performed whenever two or more tables are listed in a SQL statement.

**CONCLUSION:**

The SQL JOINS are successfully executed on the given relations.