Indian Institute of Information Technology Sri City
Database Management Systems LAB-04
Date: 17/09/2020
TOPIC : Date functions, Set operations, Subqueries(alias,set membership,set comparison,Test for empty relations)
and Basic queries on joins
Instructors: Dr. Odelu Vanga, Sarvani CH, Rangachary K, Ravi G

---

create table datedemo(val datetime);

1. **CURRENT_DATE** Returns the current date
   SELECT CURRENT_DATE();
   insert into datedemo values(current_date());

2. **CURRENT_TIME** Returns the current time
   SELECT CURRENT_TIME();
   insert into datedemo values(current_time());

3. **CURRENT_TIMESTAMP** Returns the current date and time
   SELECT CURRENT_TIMESTAMP();
   insert into datedemo values(current_timestamp());

4. **DATEDIFF** Returns the number of days between two date values
   SELECT DATEDIFF("2017-06-25 09:34:21", "2017-06-15 15:25:35");
   10

5. **ADDDATE** Adds a time/date interval to a date and then returns the date

   ADDDATE(*date*, INTERVAL *value addunit*)
   ADDDATE(*date*, *days*)

   SELECT ADDDATE("2017-06-15 09:34:21", INTERVAL 3 HOUR);
   SELECT ADDDATE("2017-06-15", INTERVAL -2 MONTH);

6. **ADDTIME** Adds a time interval to a time/datetime and then returns the time/datetime

   ADDTIME(*datetime*, *some_addtimevalue*)

Add 5 days, 2 hours, 10 minutes, 5 seconds, and 3 microseconds to a time and return the datetime:
SELECT ADDTIME("2017-06-15 09:34:21.000001", "5 2:10:5.000003");

Set operations:

customer_name	account_number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

union	**Union all**	**intersect**	**Intersect all**	**except**	**Except all**
1. Johnson 2. Hayes 3. Smith 4. Jones 5. Lindsay 6. Turner 7. Adams 8. Williams 9. Curry	1. Johnson 2. Hayes 3. Johnson 4. Smith 5. Jones 6. Lindsay 7. Turner 8. Smith 9. Hayes 10. Adams 11. Jones 12. Williams 13. Smith 14. Curry	1. Hayes 2. Smith 3. Jones	1. Hayes 2. Smith 3. Jones	1. Johnson 2. Lindsay 3. Turner	1. Johnson 2. Johnson 3. Lindsay 4. Turner

Schema for the following examples

small_customers
id
name
age
address
salary

Orders
oid
date
customer_id
amount

1. create table small_customers(id smallint,name varchar(10),age smallint,address varchar(15),salary int);
2. create table small_customers2(id smallint,name varchar(10),age smallint,address varchar(15),salary int);
3. create table orders (oid int,date datetime,customer_id smallint,amount int);
4. LOAD DATA LOCAL INFILE 'small_customers.csv' INTO table small_customers COLUMNS TERMINATED BY ',';
5. LOAD DATA LOCAL INFILE 'orders.csv' INTO table orders COLUMNS TERMINATED BY ',';

SUB QUERIES:

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

With select:

```
SELECT * FROM small_customers
WHERE ID IN (SELECT ID FROM small_customers WHERE SALARY > 4500) ;
```

With insert:

```
INSERT INTO small_customers2
SELECT * FROM small_customers
WHERE ID IN (SELECT ID FROM small_customers) ;
```

With update:

```
UPDATE small_customers
SET SALARY = SALARY * 0.25
WHERE AGE IN (SELECT AGE FROM small_customers2 WHERE AGE >= 27 );
```

With delete:

```
DELETE FROM small_customers
WHERE AGE IN (SELECT AGE FROM small_customers2 WHERE AGE >= 27 );
```

CARTESIAN PRODUCT:

Cartesian product:

```
SELECT  ID, NAME, AMOUNT, DATE FROM small_customers, orders;

SELECT  count(*) FROM small_customers, orders;
```

- alter table orders change column oid id smallint;

```
Select small_customers.id,name,orders.id
from small_customers, orders;
```

Rename operation:

column

```
Select small_customers.id as customer_id,name,orders.id as order_id from small_customers,
orders;
```

table

```
Select s.id,name,o.id
from small_customers as s, orders as o;
```

Set comparison:

The ANY and ALL operators are used with a WHERE or HAVING clause.

1. **ALL:** The ALL operator returns true if all of the subquery values meet the condition.
   ```
   SELECT id from small_customers WHERE id <> ALL(Select customer_id from orders);
   ```
2. **ANY:**The ANY operator returns true if any of the subquery values meet the condition.

```
SELECT id from small_customers WHERE id = ANY(Select customer_id from orders);

SELECT id from small_customers WHERE id = SOME(Select customer_id from orders);
```

Test for empty relations:

```
Select Name from small_customers

WHERE NOT EXISTS (SELECT * FROM orders

                  WHERE small_customers.id = orders.customer_id);
```

JOINS:

- ➢ alter table orders change column id oid smallint;
- ➢ alter table orders change column customer_id id smallint;

Natural join:

select orders.id,name from small_customers **natural join** orders

Inner join:

- **on/using**

1. select orders.id,name from small_customers **inner join** orders **on** small_customers.id=orders.id;

2. select orders.id,name from small_customers **inner join** orders **using**(id)

Right outer join:

- **Natural**
- **on/using**

1. select orders.id,name from small_customers **right join** orders **on** small_customers.id=orders.id;
2. select orders.id,name from small_customers **right join** orders **using**(id);
3. select orders.id,name from small_customers **natural join** orders;

Left outer join:

- **Natural**
- **on/using**

select orders.id,name from small_customers left join orders on small_customers.id=orders.customer_id;

Practice questions:

1. Find all the bank customers having a loan,an account or both at the bank
2. FInd those customers who are borrowers from the bank and who appear in the list of account holders (i.e present in depositor table)
3. Find all the customers who have loan at the bank ,but do not have an account at the bank
4. Find the names of all branches that have assets greater than those of atleast one branch located in Brooklyn (without using subquery)
5. Find the names of all branches that have assets greater than those of atleast one branch located in Brooklyn (using subquery)
6. Find the branch that has the highest average balance
7. Find all the customers who have both an account and a loan at the bank,by a subquery using “exists” key word
8. Perform natural join between tables loan and borrower
9. Perform inner join between tables loan and borrower,with loan_number as joining condition
10. Perform natural right outer join between tables loan and borrower
11. Perform right outer join between tables loan and borrower,with loan_number as joining condition
12. Perform natural left outer join between tables loan and borrower
13. Perform left outer join between tables loan and borrower,with loan_number as joining condition
14. Perform full outer join