4. Advanced SQL

www.LearnDB.com

Dr. Imed Bouchrika
Dept of Mathematics & Computer Science
University of Souk-Ahras
imed@imed.ws

SQL

- SQL = Structured Query Language
- The ANSI standard language for the definition and manipulation of ***relational database***.
- Includes data definition language (DDL), statements that specify and modify database schemas.
- Includes a data manipulation language (DML), statements that manipulate database content

SQL Basic Commands

- CREATE TABLE ….
- INSERT INTO clients (col1, col2) VALUES (‘val1’, ‘val2’)
- DELETE FROM clients WHERE id=6
- UPDATE clients SET name=‘abs’ WHERE id=6
- SELECT * FROM clients WHERE id=6 OR name=‘abs%’

Aggregate Functions

- The usual SQL functions that are mostly used are listed below:
  - AVG( ) – Returns the average value
  - COUNT( ) – Returns the number of rows
  - SUM( ) – Returns the sum
  - FIRST( ) – Returns the first value
  - LAST( ) – Returns the last value
  - MAX( ) – Returns the largest value
  - MIN( ) – Returns the smallest value

Aggregate Functions

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

  - Select max(price) from products
  - Select count(*) as cc from clients
  - Select avg(price) from products
  - Select sum (price) from orders where clients_id=6

Group BY

- The SQL GROUP BY clause is used with the SELECT statement to arrange identical data into groups.

- Syntax:

```
SELECT column1, column2
FROM table_name
WHERE [ conditions ]
GROUP BY column1, column2
ORDER BY column1, column2
```

- Note that **aggregate functions** are used most of the time with GROUP BY

Group BY

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- Select countries_id, count(id)
  from clients group by countries_id

- Select clients_id, sum(price) as cc
  from orders group by clients_id

HAVING

- The HAVING statement is added to SQL because the WHERE keyword could **not be** used with aggregate functions (such as count, max …)

- Syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE conditions
GROUP BY column_name
HAVING aggregate_function(column_name) operator value;
ORDER BY column_name
```

HAVING

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the customers who have over 16 orders ?**

```
SELECT clients.name,clients.last_name,count(*) as total
from clients, orders
where clients.clients_id=orders.clients_id
group by clients.clients_id
having count(*) > 16
order by total DESC
```

HAVING

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the first five countries with the largest number of orders WITH a total revenue over 11200?**

```
SELECT countries.country, count(*) as tot, sum(price) as rev
from countries, clients, orders
where clients.clients_id=orders.clients_id and
clients.countries_id=countries.countries_id
group by clients.countries_id
having sum(orders.price) > 11200
order by tot  DESC limit 0,5
```

SQL Functions

- **UCASE( )**
  converts a text to upper case letters.

- **LCASE ( )**
  converts a text to lower case letters

- **LENGTH( )**
  return the number of characters within a text.

- **ROUND( )**
  rounds a decimal number to an integer.

SQL Functions

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- Select * from products where LCASE(name) like ‘apple’
- Select ROUND(SUM(price)) as cc from orders
- Select * from clients where LENGTH(name)=3
- Select * from products where soundex(name) like soundex(‘java’)

Nested Queries

- A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.
- 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.
- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like
  - **=, <, >, >=, <=, IN, BETWEEN etc.**
- Subqueries are most frequently used with the SELECT statement.

Nested Queries

- Syntax:

```
SELECT column_name
FROM table1
WHERE column_name OPERATOR
(SELECT column_name FROM table3 WHERE condition)
```

Nested Queries

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the customers who are from Albania and Algeria?**

```
SELECT clients.name,clients.last_name
from clients
where
        clients.countries_id in
        (   select countries_id from countries where country like
            'Albania' or country like 'Algeria‘  )
```

Nested Queries

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the total payments for every customer from Canada?**

```
SELECT clients.name, clients.last_name,sum(orders.price)
from orders, clients
where orders.clients_id=clients.clients_id and
clients.countries_id=(
    select countries_id from countries where country like 'Canada‘
    )
group by clients.clients_id
```

Natural JOIN

- The JOIN is used in an SQL statement to query data from two or more tables, based on a **relationship between** certain columns in these tables.

- Syntax

```
select table1.colA, table2.colB
from table1 , table2
where
table1.colA=table2.colC
```

Natural JOIN

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Display Customer name along with their country name**

SELECT clients.name, countries.country
from clients, countries
where
clients.countries_id = countries.countries_id

LEFT JOIN

- The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2).

- The **result is NULL** in the right side when there is no match.

- Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

- OR

```
SELECT column_name(s)
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

LEFT JOIN

- Examples:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Display Customer name along with their country name**

```
SELECT clients.name, countries.country
from clients
        LEFT JOIN countries ON
        clients.countries_id =  countries.countries_id
where 1
```

UNION

- The SQL UNION operator combines the result of two or more SELECT statements.
- Each SELECT statement within the UNION must have the same number of columns.
- The columns must also have similar data types.
- The UNION operator selects only distinct values by default.
- To allow duplicate values, use UNION ALL

UNION

- Syntax:

  SELECT column_name(s) FROM table1

  UNION

  SELECT column_name(s) FROM table2;

- OR

  SELECT column_name(s) FROM table1

  UNION ALL

  SELECT column_name(s) FROM table2;

UNION

- Example:

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

INTERSECT

- This INTERSECT returns only common rows returned by the two SELECT statements.

- The same rules for UNION apply to the INTERSECT operator.

- MySQL does not support INTERSECT operator

- Syntax:

SELECT column1 [, column2] FROM table1 [, table2]
[WHERE condition]
INTERSECT
SELECT column1 [, column2] FROM table1 [, table2]
[WHERE condition]

Triggers

- a **trigger** is a **rule** that you put on a table which basically says, whenever you DELETE, UPDATE or INSERT something in this table, also do something else.

- Syntax:

```
CREATE
   TRIGGER event_name  BEFORE/AFTER    INSERT/UPDATE/DELETE
   ON database.table
            FOR EACH ROW BEGIN
                     -- trigger body
                     -- this code is applied to every
                     -- inserted/updated/deleted row
   END;
```

Triggers :: Example

```
DELIMITER $$
CREATE
    TRIGGER  blog_after_update AFTER UPDATE   ON  blog
    FOR EACH ROW BEGIN

        IF NEW.deleted THEN
                SET @changetype = 'DELETE';
        ELSE
                SET @changetype = 'EDIT';
        END IF;

        INSERT INTO audit (blog_id, changetype) VALUES (NEW.id, @changetype);
  END$$
DELIMITER ;
```

Triggers :: Example

```
DELIMITER $$
CREATE

    TRIGGER  check_amount BEFORE UPDATE   ON  account
    FOR EACH ROW BEGIN

        IF NEW.amount<0 THEN
                NEW.amount=0;
        END IF;

      END$$
DELIMITER ;
```

Foreign Key

- A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

- Syntax :

```
create table orders (
  orders_id int not null auto_increment primary key,
  price double not null ,
  clients_id int not null,
  products_id int not null,
  FOREIGN KEY clients_id REFERENCES clients(clients_id)
    ON DELETE CASCADE  ON UPDATE CASCADE,
  FOREIGN KEY products_id REFERENCES products(products_id)
)
```

Foreign Key

Constraint	Description
ON DELETE CASCADE	When a row in the parent table is deleted, InnoDB will automatically delete corresponding foreign key column in the child table.
ON DELETE SET NULL	When a row in the parent table is deleted, InnoDB will automatically set corresponding foreign key column in the child table to **NULL**.
ON DELETE RESTRICT	ON DELETE RESTRICT disallows a delete if an associated record still exists.
ON UPDATE CASCADE	→ update corresponding foreign key column in all matching rows in the child table to the same value.
ON UPDATE SET NULL	→ set corresponding foreign key column in all matching rows in the child table to NULL.
ON UPDATE RESTRICT	ON UPDATE RESTRICT disallows an update if an associated record still exists.

Examples

- Database Structure:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the total revenue for the months of Junes (ALL Years) ?**

```
SELECT sum(price) as revenue
FROM orders
where o_time like  '____-06-__'
```

Examples

- Database Structure:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the three most ordered products ?**

```
SELECT products.name,  count(*) as total
from products, orders
where products.products_id=orders.products_id
group by products.products_id
order by total DESC Limit 0, 3
```

Examples

- Database Structure:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the client having paid the largest amount ?**

```
SELECT clients.name, sum(price) as total
from clients, orders
where
clients.clients_id=orders.clients_id
group by clients.clients_id
order by total  DESC limit 0,1
```

Examples

- Database Structure:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the most popular product in Canada for the year 2012 ?**

```
SELECT products.name, countries.country,count(*) as total
from countries, clients, orders,products
where
clients.clients_id=orders.clients_id and
clients.countries_id=countries.countries_id and
products.products_id=orders.products_id and
o_time like '2012-%' and
countries.country = 'Canada'
group by orders.products_id order by total  DESC limit 0,1
```

Examples

- Database Structure:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Display Customer name along with their country name**

```
SELECT clients.name, countries.country
from clients
      LEFT JOIN countries ON
      clients.countries_id =  countries.countries_id
where 1
```

Examples

- Database Structure:

clients(clients_id, countries_id, name, last_name, phone)
countries(countries_id, country)
products(products_id, name, price)
orders(orders_id , clients_id , products_id, price, o_time)

- **Find the first five countries with the largest number of orders ?**

```
SELECT countries.country, count(*) as total
from countries, clients, orders
where
clients.clients_id=orders.clients_id and
clients.countries_id=countries.countries_id
group by clients.countries_id
order by total  DESC limit 0,5
```

For you to search !

- **MySQL Procedures**.
- **InnoDB**