SQL
For QA Engineers

PART 2

Good To Know

- SQL is NOT case-sensitive.

  Can be lower or uppercase, but often keywords are written in all cap

- Numeric values should not be enclosed in quotes, but alphanumeric should be in single quote

  SALARY = 5000

  LAST_NAME = 'Smith'

How to read data?

Now, that we learned how to create, modify, and delete database tables, how to insert, update and delete data from those tables our next goal is to QUERY (read) the tables data.

How?

SELECT Statement

To retrieve data from database (Tables) we use the SELECT statement:

SELECT [ALL | DISTINCT] c*olumn1*[,*column2*]
FROM <table1> [,<*table2*>]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC]]

All values in [ ] are optional for the SELECT

SELECT???? What is it?

SELECT Statement

To retrieve data from database (Tables) we use the SELECT statement:

SELECT [ALL | DISTINCT] ***column1*** **[,*column2*]**
FROM <table1> [,<*table2*>]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC]]

Let's start with the SELECT and FROM keywords

SELECT…FROM

SELECT and FROM are the only two required keywords

SELECT:

- *Specifies what data to retrieve;*
- *Minimum of one column, but as many as we need;*
- *All selected columns should come from the table(s), specified in FROM.*

FROM:

- *Specifies where is it coming from;*
- *Minimum of one table, but as many as exist in dB;*

SELECT Column(s)

SELECT [ALL | DISTINCT] *column1* [,*column2*]

To select ALL columns:

SELECT * FROM <table name>

The asterisk (*) is a quick way of selecting all columns!

To select SPECIFIC columns from tables:

SELECT *<column1>, <column2>*
FROM <table name>

To eliminate duplicate values from a column:

SELECT DISTINCT <column1>
FROM <table name>

Exercise – SELECT

- Type the attached commands
- Highlight one statement at a time and RUN the queries

1) Select ALL Columns from the Table:

```
SELECT * FROM EMPLOYEES
```

2) Select SPECIFIC Columns from the table:

```
SELECT FIRST_NAME, LAST_NAME
FROM   EMPLOYEES
```

What columns returned first query? Second? Why?

3) Select ALL VALUES in column (including duplicates):

```
SELECT DEPARTMENT_ID FROM EMPLOYEES
```

4) Select UNIQUE VALUE in column (NO duplicates):

```
SELECT DISTINCT DEPARTMENT_ID FROM EMPLOYEES
```

Quiz

Table: DEPARTMENTS

1) Read all information about departments

2) Show only departments id and name

3) Display all location ids

4) Hoops! Please display each location id just once

5) Display all unique department ids with location ids

SELECT Statement

To retrieve data from database (Tables) we use the SELECT statement:

SELECT [ALL | DISTINCT] c*olumn1* [,*column2*]
FROM <table1> [,<*table2*>]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC]]

Optional, but very important is WHERE clause.

SELECT…WHERE

So far we learned how to read the data for the *whole table*.

For example:

```
select department_id, department_name
from  departments
```

What if we need *only some data*?

For example:

Find the department name for department_id "80":

```
select department_id, department_name
from  departments
where department_id=80
```

SELECT…WHERE

WHERE allows us to specify a condition as:

- Comparison: =, >, <, >=, <=
  - WHERE DEPARTMENT_ID > 80
  - WHERE DEPARTMENT_ID <> 80
  - WHERE DEPARTMENT_ID <= 80
- Negative Conditions: <> (Not equal), NOT, etc
  - WHERE DEPARTMENT_ID <> 80
  - WHERE NOT DEPARTMENT_ID = 80

Exercise – SELECT…WHERE

Run the following commands:

```
select  department_id, department_name
from    departments
where department_id = 80
```

****Change = to >, <, >=, etc.*

```
select  department_id, department_name
from    departments
where department_id <> 80
```

```
select  department_id, department_name
from    departments
where NOT department_id = 80
```

Quiz

Table: EMPLOYEES

1) Read first name, last name and department id for all employees, who are working in the department 80

2) Read first name, last name and salary for all employees, who are making over $10,000

3) Read first name, last name and department id for all employees, who are:
   - working on commissions over 0.3%;
   - working on commissions not over 0.3%;

SELECT…WHERE

WHERE allows us to specify more than one condition:

- Boolean: AND, OR
  - WHERE DEPARTMENT_ID > 80 AND LAST_NAME = 'Smith'
  - WHERE DEPARTMENT_ID = 80 OR DEPARTMENT_ID = 100

AND:

- Returns rows for which *both* conditions are true;

OR:

- Returns rows for which *either* condition is true.

Parentheses around the statements are optional, but may insure they are evaluated as a group.

Exercise – SELECT…WHERE

Run the following commands:

```
select  *
from    employees
where   department_Id = 110
or      department_Id = 100

select  *
from    employees
where   department_Id = 100
and     salary > 8000

select  *
from    employees
where   department_Id = 100
or      salary > 8000
```

Quiz

Table: EMPLOYEES

1) Read all department 80 employees, who are making over $10,000 (ids are between 100 and 150);

2) In the departments 30 and 90 find all employees with the last name 'King';

3) Display employee, if his last name is "King" or if he works in the departments 30 or 90

SELECT…WHERE

1) Read all employees, from the departments 30, 40, 50, 60, 70, 80, 90, 100, 110;

2) Read all employees, whose salary is between $12,000 and $15,000;

3) Read all employees, whose name starts with ‘A’

How do we do that?

SELECT…WHERE

We learned the following operators:

- Comparison: =, >, <, >=, <=, <>
- Boolean: AND, OR, NOT

WHERE let us specify more complicated operators:

- IN
- BETWEEN
- LIKE
- IS NULL / IS NOT NULL
  - WHERE DEPARTMENT_ID IN (80, 90, 100)
  - WHERE DEPARTMENT_ID LIKE "LIN%"
  - WHERE DEPARTMENT_ID IS NOT NULL

Operator IN, BETWEEN

IN - specify multiple values in WHERE clause.

For Example:

Read all employees, from the departments 30, 40, 50, 60, 70):

```
select *
from employees
where department_id in (30, 40, 50, 60, 70)
```

BETWEEN - select a range of data between two values.

For Example:

Read all employees, whose salary is between $12,000 and $15,000:

```
select *
from employees
where salary between 12000 and 15000
```

Operator LIKE

LIKE operator allows us to search for a specified pattern in a column.

```
SELECT   EMPLOYEE_ID
FROM     EMPLOYEES
WHERE    FIRST_NAME LIKE 'Nan%'
```

% sign can be used to define missing character(s) in the pattern both before and after the pattern.

_ (underscore) sign can be used to define only ONE missing character in the pattern both before and after the pattern

- FIRST_NAME LIKE 'Nan%' Nan, Nancy, Nanzan, Nanitta
- FIRST_NAME LIKE '%Nan' A'Nan, BeNan
- FIRST_NAME LIKE '%Nan%' Nan, A'Nan, BeNaner
- FIRST_NAME LIKE '_ean' Dean, Sean, Peancy

Operator IS NULL, IS NOT NULL

Since NULL is just an unknown data (placeholder), we can not use regular operators: =, >, <, etc.

First Name	Last Name	Address
Suman	Singh	1221 Main Street, Los Altos, CA 94352
Boris	Beyer	

Only operators IS NULL or IS NOT NULL

Quiz

1) In which department(s) we have employees, working on commissions;

2) Retrieve all employees, who are working on commission;

SELECT Statement

To retrieve data from database (Tables) we use the SELECT statement:

SELECT [ALL | DISTINCT] c*olumn1*[,*column2*]
FROM <table1> [,<*table2*>]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC]]

So far we discussed SELECT, FROM and WHERE. Let's check the ORDER BY.

SELECT…ORDER BY

ORDER BY sorts my results in a specific order

```
SELECT     LAST_NAME, FIRST_NAME
FROM       EMPLOYEES
ORDER BY LAST_NAME [ASC | DESC]
```

- ASC (default) - small to large (1, 2, 3, 4, 5)
- DESC - large to small (5, 4, 3, 2, 1)

If you do not list sort order, it will be sorted in ASCending (default) order.

Quiz

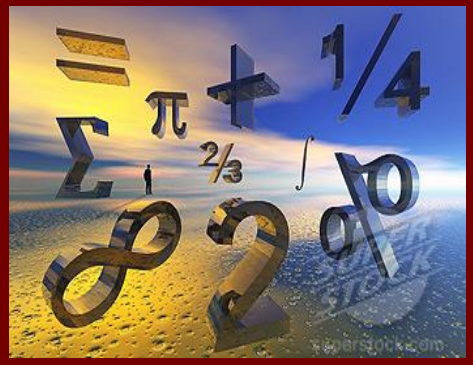
1) Read employees info (salary, last_name, department_id), showing the highest paid first;

2) Provide the complete list of all employees within the department (show the departments in order and then last_names in order within a department);

Aggregate Functions

We learned how to query (read) data from the table. But I want the computer to do some calculation for me as well.

- How many employees are in my company?
- What is the highest and lowest salary?
- What is an average salary?



Aggregate functions return a single value, calculated from values in a column, do some calculation for us.

Useful Aggregate Functions

Work with any data type (numeric, character or date)

- COUNT() - Returns the number of rows
- MAX() - Returns the largest value
- MIN() - Returns the smallest value

Work ONLY with numeric VALUES

- SUM() - Returns the sum
- AVG() - Returns the average value

Exercise

1) How many employees are working in the department 100?

```
select count(*)
from   employees
where department_id = 100
```

2) What is the min, max, average, total salary for the department 80?

```
select  min(salary), max(salary), avg(salary), sum(salary)
from   employees
where department_id = 80
```

Confirm:

```
select salary
from employees
where department_id = 80
order by salary
```

Quiz

1) What is the total commission percent for all employees within the company?

2) If you'll sort employees last names in the ascending order, what will be the first and last on your list?

Advanced:

3) How many people work in each department?

REVIEW!

- Database (dB) is a storage space for content / information (data);
  It is an organized collection of data;
- Relational databases use tables to store information and specify the relations between those tables;
- Schema is a logical container for database objects (tables, views, triggers) that user creates;
- Table is a container for data elements and relations. It's using columns (fields) and rows (records);
- Field is a database storage simplest unit (table column);
- Record is a row and it represents a single structured data set in table;
- Query is your request to the database to retrieve *information;*
- Report is the returned information to the specified query;
- DBMS (Database Management System) is a software that controls the organization, storage, retrieval, security and integrity of data in dB;
- Popular Databases: ORACLE, Sybase, MySQL, DB2, etc.

- PRIMARY KEY (PK) is a *unique identifier* of every record in database;
- FOREIGN KEY (FK) is a column (or combination of columns) that is used to establish a relationship between the tables; Foreign key is usually not unique (one-to-many relation) and shall always point to a primary key.
- NORMALIZATION is the dB process of organizing the fields and tables of a relationship database to minimize redundancy and dependency. Divide large tables into smaller tables and defining relationships between them;
- SQL stands for Structured Query Language. It is a database computer language, designed to retrieve and manage data, create and modify dB schema, etc.
- SQL Commands:
  - Create, modify, delete, and view tables (CREATE, ALTER, DROP (delete table), DESC)
  - Populate tables with data (INSERT, DELETE, TRUNCATE (delete data, but not table), etc)
  - Query the table(s) data (SELECT)

DDL (Data Definition Language) statements are defining database schema:

- CREATE - to create objects in the database (TABLE, VIEW, INDEX)
- ALTER - alters the structure of the database (TABLE, VIEW, INDEX)
- DROP - delete objects from the database (TABLE, VIEW, INDEX)
- TRUNCATE - remove all records from a table
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

DML (Data Manipulation Language) statements are managing data within schema:

- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all or specific (+WHERE) records from a table

DCL (Data Control Language) statements:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

CREATE - creates a table (index, view) in a database.

```
CREATE TABLE table1 (ID int, Last_Name varchar(255))
```

ALTER - used to add, delete, or modify columns in an existing table.

```
ALTER TABLE table1 ADD column1 varchar(255)
ALTER TABLE table1 DROP COLUMN column1
ALTER TABLE table1 MODIFY column1 varchar(255)
```

DROP – deletes a table (index, view).

```
DROP TABLE table1
```

TRUNCATE - deletes data in the table, but not table itself

```
TRUNCATE TABLE table1
```

INSERT - inserts record into a table (don't violate constraints)

```
INSERT into  table1 VALUES (value1, value2, etc.)
```

UPDATE - modifies record in a table (no primary keys update)

```
UPDATE  table1
SET column1 = 'Smith'
WHERE   column1 = 'Allins'
```

DELETE - deletes record from a table (**all** records, if there is no WHERE)

```
DELETE FROM table1
[WHERE   column1 = 'Allins']
```

SELECT - retrieves data from one or more dB tables or views. Most often used statement

SELECT [ALL | DISTINCT] c*olumn1*[,*column2*]
FROM <table1> [,*<table2>*]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC]]

1. SELECT… FROM… are the **only two required keywords**;
2. SELECT * … - returns **ALL columns**;
3. SELECT <column(s)> - returns **specified columns**;
4. SELECT DISTINCT <column1>… - returns **unique column** values only;
5. SELECT ROWNUM, … - returns **numbers of** each returned **row**;
6. Numeric values should not be enclosed in quotes, but text values should: (SALARY > 200 and NAME = 'Smith')
7. FROM - must contain **at least one table** from where we are retrieving information;
8. WHERE, GROUP BY, HAVING, ORDER BY-are **optional**, but must be in the **specific order**;
9. To select ALL TABLES: select * from all_tables

SELECT [ALL | DISTINCT] *column1*[,*column2*]
FROM <table1> [,<*table2*>]
[WHERE "conditions"]
[GROUP BY "column-list"]
[HAVING "conditions]
[ORDER BY "column-list" [ASC | DESC]]

- WHERE – optional. If used, specify any condition(s) by using operators:

  =, >, <, >=, <=, <> (not equal);
  AND (both conditions),
  OR (one or another, or both);
  IN (specify multiple values);
  BETWEEN (select a range of data between two values);
  LIKE (search for a specified pattern in a column: % or _ (one character))
  IS NULL / IS NOT NULL (operator for an unknown (NULL) value)

  Examples:

  ```
  WHERE (DEPARTMENT_ID > 50 AND <> 100) OR (DEPARTMENT_ID = 20 )
  WHERE DEPARTMENT_ID IN (80, 90, 100)
  WHERE DEPARTMENT_ID BETWEEN 80 AND 120
  WHERE DEPARTMENT_ID LIKE 'LIN%
  WHERE DEPARTMENT_ID IS NOT NULL
  ```

- ORDER BY – optional, allow to order the results in specific order (ASC, default, sorts in A-Z, DESC sorts in Z-A)

SQL Part2
QUIZ

Use EMPLOYEES table for all tasks:

1) Get the list of names and department Ids of everybody, reporting to manager (ID=103), or working at the Department 80, 60, or 90; Sort retrieved names within the department, having the largest department on top;

2) Find out how much Luis Popp and David Austin are making a month?

3) Find Ki Gee phone number;

4) Provide all information on employees, working on commission basis;

5) Find all managers with id 108 or unknown, making more than $7,700 a month;

6) Display all unique numeric department ids, sorted by highest department number on top;

7) Please select all information from table “Employees” where last name contains “or”;

8) Our company is running a special promotion for all employees with last name starting with ‘D’. To qualify, you need to work in the department 50, 60, 90, or 100, make over $3000, and start working for our company from Jan 1, 1989 through Jan 1, 2002.

9) Company is planning to celebrate the 1st employee. When s/he was hired?

10) Get list of departments and number of employees in each department, ordered alphabetically.

Interview Questions

1. What is Constraint? Give couple constraints examples?
2. How to read data from dB?
3. What are the minimum requirements for the SELECT statement?
4. Is WHERE required in the SELECT statement? Why?
5. How can I retrieve information for the whole table?
6. How would you read all first names from CUSTOMER table?
7. Is it possible to have duplicate first names in CUSTOMER table? How would you read only unique names?
8. How would you retrieve all employees 55+ years old, living in cities, starting with 'San', showing the oldest employee first?
9. How would you read all employees who was born in Sep, Oct, and Nov?

Interview Questions

1. How would you choose all employees who do NOT have the address, stored in the system?
2. GROUP BY, HAVING, ORDER BY are required. True or False? Why?
3. How can I group results by specified criteria?
4. When are we using HAVING option?
5. What is an aggregate function? Examples?
6. How can I count all records within a table?
7. How can I count all the records within the department?
8. How can I find the average salary?

SQL Part2 Homework

1. Complete Class Exercises;
2. Review SQL Part2 Material;