JDBC Example

```
Connection conn;
// Insert code here to connect conn to a DB.
// Requires JDBC driver; See sample code in Project 2 for
// Oracle or for Sqlite, Sample.java

String q = "SELECT name FROM Students WHERE GPA > 3.5";
try {
     Statement st = conn.createStatement();
     ResultSet rs = st.executeQuery(q);

     while (rs.next()) {
        String name = rs.getString("name");
        System.out.println(name);
     }
     rs.close();
     st.close();
}
catch(SQLException e){System.err.println(e.getMessage());}
```

Cursor retrieves rows from result one at a time

Full Javadoc for java.sql available online:
http://download.oracle.com/javase/6/docs/api/

Connections

- Get Connection object:
  - Oracle requires passwords
  - Sqlite3 is file-based and does not require a password
    - Connection conn = ….
- Always close connections before quitting the program
  - conn.close();
  - Similarly, close other Oracle resources.

Cool Trick to Auto-Close Resources

- A cool trick in Java/JDBC to auto-close database connection and other resources automatically.

http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/jdbc_41.html

JDBC – AutoClose Trick

```
Connection conn;
// Obtain a connection to DB, store in conn
// (Requires JDBC driver; See sample code in Project 2)

String q = "SELECT name FROM Students WHERE GPA > 3.5";
try (Statement st = conn.createStatement()) { // auto-close
     ResultSet rs = st.executeQuery(q);

     while (rs.next()) {
        String name = rs.getString("name");
        System.out.println(name);
     }
     // rs.close();    Not needed.
     // st.close();   Not needed.
}
catch(SQLException e){System.err.println(e.getMessage());}
```

Full Javadoc for java.sql available online:
http://download.oracle.com/javase/6/docs/api/

Challenges

- DBMS and PL implement different data types
  - “Impedance Mismatch”
- Need to match DB types with PL types, e.g.,

SQL Type	Java Type	ResultSet Method
CHAR	String	getString()
VARCHAR	String	getString()
DOUBLE	Double	getDouble()
INTEGER	Integer	getInt()

JDBC Example

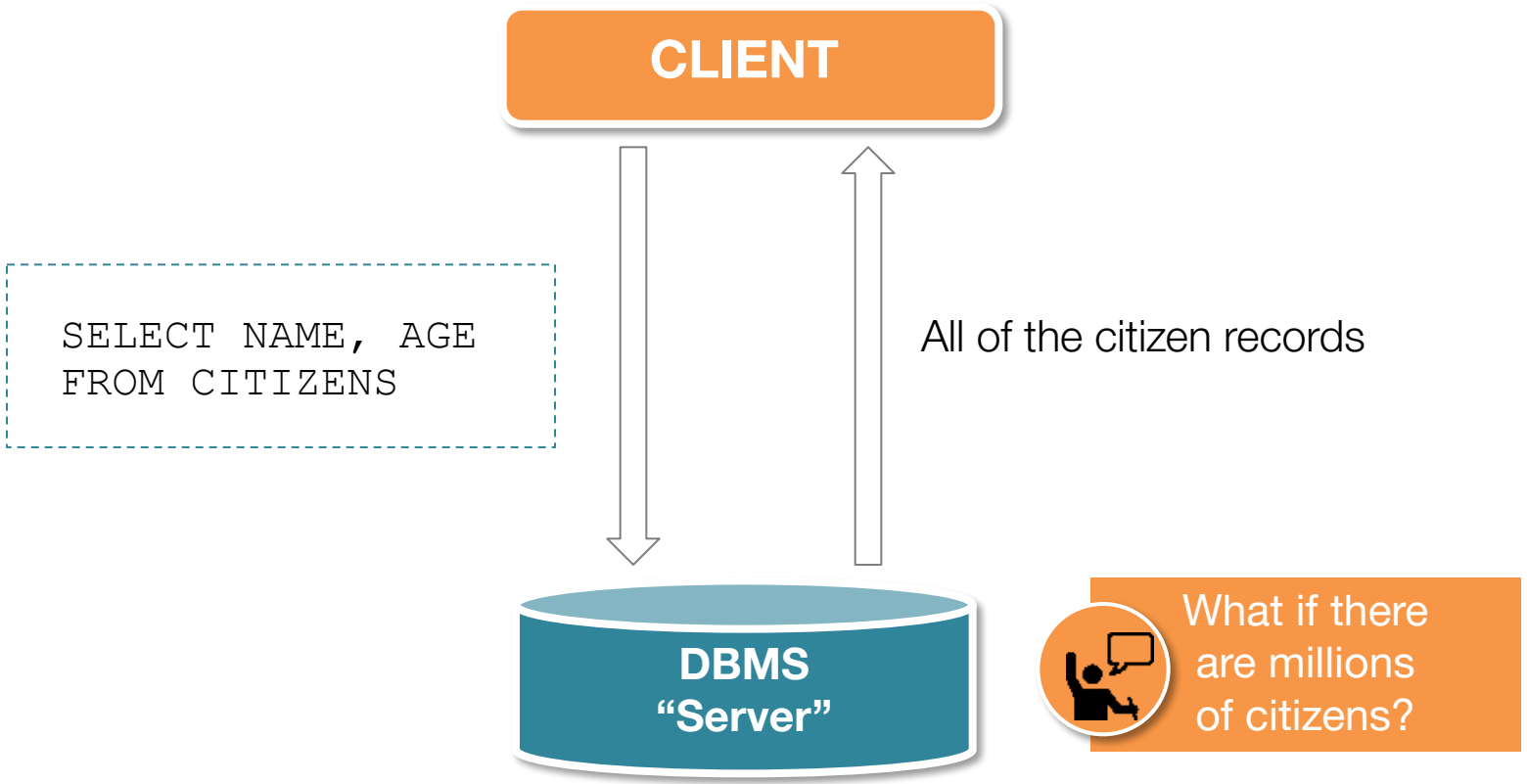
What does the following code snippet do?

```
String query = "SELECT NAME, AGE FROM CITIZENS";
try (Statement st = conn.createStatement()){ // auto-close
      double sum = 0;
      double count = 0;

     ResultSet rs = st.executeQuery(query);
     while (rs.next()) {
        String name = rs.getString("NAME");
        sum += rs.getDouble("AGE");
      count++;
     }
      System.out.println(sum/count);
}
catch(SQLException e){System.err.println(e.getMessage());}
```

What Happens?

Compute the average age



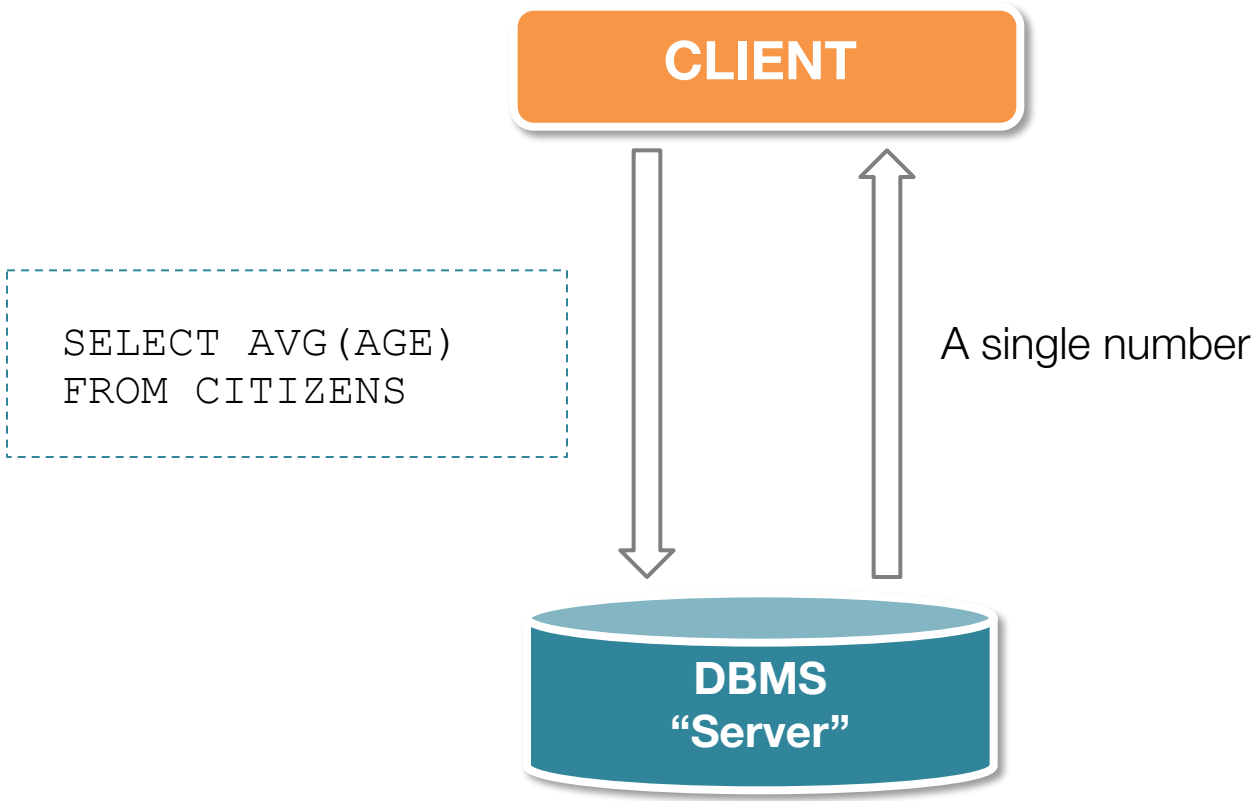
JDBC Example (Revised)

Push the computation “closer” to the data…
Make DBMS do the processing that it does well.

```
String query = "SELECT AVG(AGE) FROM CITIZENS";
Try (Statement st = conn.createStatement()) { // auto-close
     ResultSet rs = st.executeQuery(query);
     while (rs.next()) {
        Double avg = rs.getDouble(1);
          System.out.println(avg);
     }
}
catch(SQLException e){System.err.println(e.getMessage());}
```

What Happens Now?

Compute the average age



Question??

Does the above apply even if there is no aggregation?

Consider the following:

```
SELECT NAME, AGE
FROM CITIZENS
```

A

```
SELECT NAME, AGE
FROM CITIZENS
WHERE SALARY < 1000
```

B

Select which query you would prefer to send in SQL (and do the rest in Java)