```
#include <iostream>
#include <occi.h>
#include <iomanip>

using oracle::occi::Environment;
using oracle::occi::Connection;
using namespace oracle::occi;
using namespace std;

int main(void)
{
    /* OCCI Variables */
    // define a reference to objects environment, connection, statement and resultset
    Environment* env = nullptr;
    Connection* conn = nullptr;
    Statement* stmt = nullptr;
    ResultSet* rs = nullptr;

    /* Used Variables */
    string user = "dbs211_";
    string pass = "3421341";
    string constr = "myoracle12c.senecacollege.ca:1521/oracle12c";
try {
        // environment scope starts
        env = Environment::createEnvironment(Environment::DEFAULT);
        // establish a connection to the Oracle server
conn = env->createConnection(user, pass, constr);


        // Report 1


        // call method createStatement() to create an statement object
        stmt = conn->createStatement("SELECT e.employeeNumber, e.firstName, e.lastName, o.phone, e.extension FROM dbs211_employees e
INNER
JOIN dbs211_offices o ON e.officeCode = o.officeCode WHERE o.city = 'San Francisco' ORDER BY e.employeeNumber");
        // store the result set
rs = stmt->executeQuery();

        cout << "------------------------- Report 1 (Employee Report) -----------------------------------" << endl;
        cout << std::left << std::setw(14) << "Employee ID" << std::setw(19) << "First Name" << std::setw(19) << "Last Name"
            << std::setw(18) << "Phone" << std::setw(11) << "Extension" << endl;
        cout << std::left << std::setw(14) << "------------" << std::setw(19) << "-----------------" << std::setw(19) <<
"-----------------"             << std::setw(18) << "----------------" << std::setw(11) << "---------" << endl;

        if (!rs->next()) {
            // if the result set is empty
            cout << "ResultSet is empty." << endl;
        }
else {
            // if the result set in not empty
do {
                    cout  << std::left  << std::setw(14)  << rs->getInt(1)  << std::setw(19)  << rs->getString(2)  <<
std::setw(19)                         << rs->getString(3) << std::setw(18) << rs->getString(4) << std::setw(11) << rs-
>getString(5) << endl;             } while (rs->next()); //if there is more rows, iterate
        }
        cout << endl;

        stmt->closeResultSet(rs);         conn-
>terminateStatement(stmt);


        // Report 2

        stmt = conn->createStatement("SELECT DISTINCT e2.employeeNumber, e2.firstName, e2.lastName, o.phone, e2.extension FROM
dbs211_employees e1 INNER JOIN dbs211_employees e2 ON e1.reportsTo = e2.employeeNumber INNER JOIN dbs211_offices o ON e2.officeCode =
o.
officeCode WHERE e1.reportsTo IS NOT NULL ORDER BY
e2.employeeNumber");         rs = stmt->executeQuery();

        cout << "------------------------- Report 2 (Manager Report) -----------------------------------" << endl;
        cout << std::left << std::setw(14) << "Employee ID" << std::setw(19) << "First Name" << std::setw(19) << "Last Name"
            << std::setw(18) << "Phone" << std::setw(11) << "Extension" << endl;
        cout << std::left << std::setw(14) << "------------" << std::setw(19) << "-----------------" << std::setw(19) <<
"-----------------"             << std::setw(18) << "----------------" << std::setw(11) << "---------" << endl;

        if (!rs->next()) {
            // if the result set is empty
            cout << "ResultSet is empty." << endl;
        }
else {
            // if the result set in not empty
            do {
                    cout  << std::left  << std::setw(14)  << rs->getInt(1)  << std::setw(19)  << rs->getString(2)  <<
std::setw(19)                         << rs->getString(3) << std::setw(18) << rs->getString(4) << std::setw(11) << rs-
>getString(5) << endl;             } while (rs->next()); //if there is more rows, iterate         }
```

```
        // resultset scope ends         stmt-
>closeResultSet(rs);         // statement
scope ends         conn-
>terminateStatement(stmt);         //
connection to the Oracle server ends
env->terminateConnection(conn);
        // environment scope ends
        Environment::terminateEnvironment(env);

    }


    catch (SQLException& sqlExcp) {
        cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
    }
return 0;
}
```