The first query counts the number of documents in the PURCHASEORDER table using the standard XQuery function fn:collection() to access the contents of the table. fn:collection() expects a path that identifies the set of XML documents to be processed. In this case, the path is prefixed with the 'protocol' oradb, indicating that the components of the path should be interpreted as DATABASE_SCHEMA and TABLE.

The second query shows how we can use predicates to restrict which documents are returned. In this case the predicate on the Reference element uniquely identifies a single document. The document is returned as an XMLType object, and the SQL Developer script output window shows this by outputting the text "(XMLTYPE)". In order to see the XML in the SQL Developer script output window, the XML must be serialized, or converted into a textual representation, using the XMLSerialize() operator. XMLSerialize() generates a textual representation of the XML and returns it as a CLOB, BLOB or VARCHAR2 data type.

The third query shows how to use XMLSerialize() to convert the document into a serialized form, in this case stored in a CLOB.

The fourth example shows how to pass multiple predicate values into the XQuery. In a real-world example these values would be supplied as bind variables rather than hard coded literals. Also note that the XQuery expression in this example terminates in the Reference element. Consequently the result consists of just the Reference element from the documents that match the supplied predicates.

The fifth query shows how to use XQuery to synthesize a new document from the documents that match the supplied predicates.

The sixth query shows how to use the columns clause of the XMLTable operator to create an inline relational view from the documents that match the supplied predicates. In this case the XQuery expression generates a result document from each of the PurchaseOrder documents that match the supplied predicates, and then the Columns clause maps elements and attributes in the result document into the columns of the in-line view. This allows a conventional relational result to be created by executing an XQuery operation on XML content.

The final query shows how XQuery can be used with relational data. In this example a join is taking place between the XML content of the PURCHASEORDER table and the content of the relational tables HR.EMPLOYEES and HR.DEPARTMENTS.