

Lab 8: Views, Scripts

1. Write a view named InvoiceBasic that returns three columns: VendorName, InvoiceNumber, InvoiceTotal. Then, write a SELECT statement that returns all the columns in the view, sorted by InvoiceTotal from smallest to largest, where the first letter of the vendor name is A, B, or C.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'DESKTOP-PTHKBA0 (SQL Server 15.0.2000.5 - D)'. The main window shows a query script for 'Query1.sql - DES...PTHKBA0\lyq (53)'. The script defines a view 'InvoiceBasic' and a query to select from it.

```
CREATE VIEW InvoiceBasic
AS
SELECT VendorName, InvoiceNumber, InvoiceTotal
FROM Vendors JOIN Invoices ON Vendors.VendorID = Invoices.VendorID;

SELECT * FROM InvoiceBasic
WHERE ( LEFT ( VendorName, 1 ) = 'A' OR LEFT ( VendorName, 1 ) = 'B' OR LEFT ( VendorName, 1 ) = 'C' )
ORDER BY InvoiceTotal DESC;
```

The Results pane shows the output of the query, displaying 12 rows of data sorted by InvoiceTotal in descending order.

	VendorName	InvoiceNumber	InvoiceTotal
1	Bertelsmann Industry Svcs. Inc.	509786	6940.25
2	Computerworld	367447	2433.00
3	Cahners Publishing Company	587056	2184.50
4	Blue Cross	547481328	224.00
5	Blue Cross	547480102	224.00
6	Cardinal Business Media, Inc.	133560	175.00
7	Blue Cross	547479217	116.00
8	Cardinal Business Media, Inc.	134116	90.36
9	Coffee Break Service	109596	41.80
10	Abbey Office Furnishings	203339-13	17.50
11	Compuserve	21-4748363	9.95
12	Compuserve	21-4923721	9.95

The status bar at the bottom indicates: Query executed successfully. DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (53) AP 00:00:00 12 rows

2. Create a view named Top10PaidInvoices that returns three columns for each vendor: VendorName, LastInvoiceDate (the most recent invoice date), and SumOfInvoices (the sum of the InvoiceTotal column). Return only the 10 vendors with the largest SumOfInvoices and include only paid invoices (i.e. InvoiceTotal - CreditTotal - PaymentTotal = 0). Then write a SELECT statement to show results of the view.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server structure for 'DESKTOP-PTHKBA0 (SQL Server 15.0.2000.5 - D)'. The central pane shows the SQL Query Editor with the following T-SQL script:

```
CREATE VIEW Top10PaidInvoices AS
SELECT TOP 10 VendorName, MAX(InvoiceDate) AS LastInvoiceDate, SUM(InvoiceTotal) AS SumOfInvoices
FROM Vendors JOIN Invoices ON Vendors.VendorID=Invoices.VendorID
WHERE InvoiceTotal - CreditTotal - PaymentTotal = 0
GROUP BY VendorName
ORDER BY SumOfInvoices DESC;

SELECT * FROM Top10PaidInvoices
```

The bottom pane shows the 'Results' tab with the following data:

	VendorName	LastInvoiceDate	SumOfInvoices
1	Malloy Lithographing Inc	2016-03-24 00:00:00	88365.17
2	United Parcel Service	2016-03-24 00:00:00	23177.96
3	Data Reproductions Corp	2016-02-01 00:00:00	21842.00
4	Digital Dreamworks	2016-01-21 00:00:00	7125.34
5	Bertelsmann Industry Svcs. Inc	2016-02-18 00:00:00	6940.25
6	Zylka Design	2016-03-25 00:00:00	6940.25
7	Yesmed, Inc	2016-01-11 00:00:00	4901.26
8	Federal Express Corporation	2016-04-02 00:00:00	4167.13
9	Computerworld	2016-02-11 00:00:00	2433.00
10	Cahners Publishing Company	2016-02-28 00:00:00	2184.50

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (68) AP 00:00:00 10 rows'.

3. Create an updatable views named VendorAddress that returns the VendorID, Address (i.e. VendorAddress1 + ' ' + VendorAddress2), and the city, state, and zipcode columns for each vendor. Whenever VendorAddress2 is NULL, substitute the NULL with a blank space. Then write a SELECT query to examine the result set where VendorID=7. Write a SELECT statement to verify the result.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'DESKTOP-PTHKBA0 (SQL Server 15.0.2000.5 - D...)'.

The central pane shows the SQL query editor with the following code:

```
CREATE VIEW VendorAddress
AS
SELECT VendorID, VendorAddress1+' '+VendorAddress2 AS 'Address', VendorCity AS 'City', VendorState AS 'State', VendorZipcode AS 'Zipcode'
FROM Vendors;

SELECT * FROM VendorAddress
WHERE VendorID = 7;
```

The bottom pane shows the results of the query execution. The status bar indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (65) AP 00:00:00 1 rows'.

VendorID	Address	City	State	Zipcode
7	Kevin Minder 3441 W Macarthur Blvd	Santa Ana	CA	92704

4. Write a SELECT statement that selects all the columns for the catalog view that returns information about foreign keys in the AP database. How many foreign key(s) is/are defined in the AP database and what is/are they?

As we could see in the screenshot, there are 6 foreign keys in the AP database, they are : InvoiceLineItems.GLAccounts, InvoiceLineItems.Invoices, Invoices.Terms, Invoices.Vendors, Vendors.GLAccount, Vendors.Terms.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the 'AP' database structure, including tables, views, and security. The main window shows a query executed in the 'AP' database, which returns a list of foreign keys. The query is as follows:

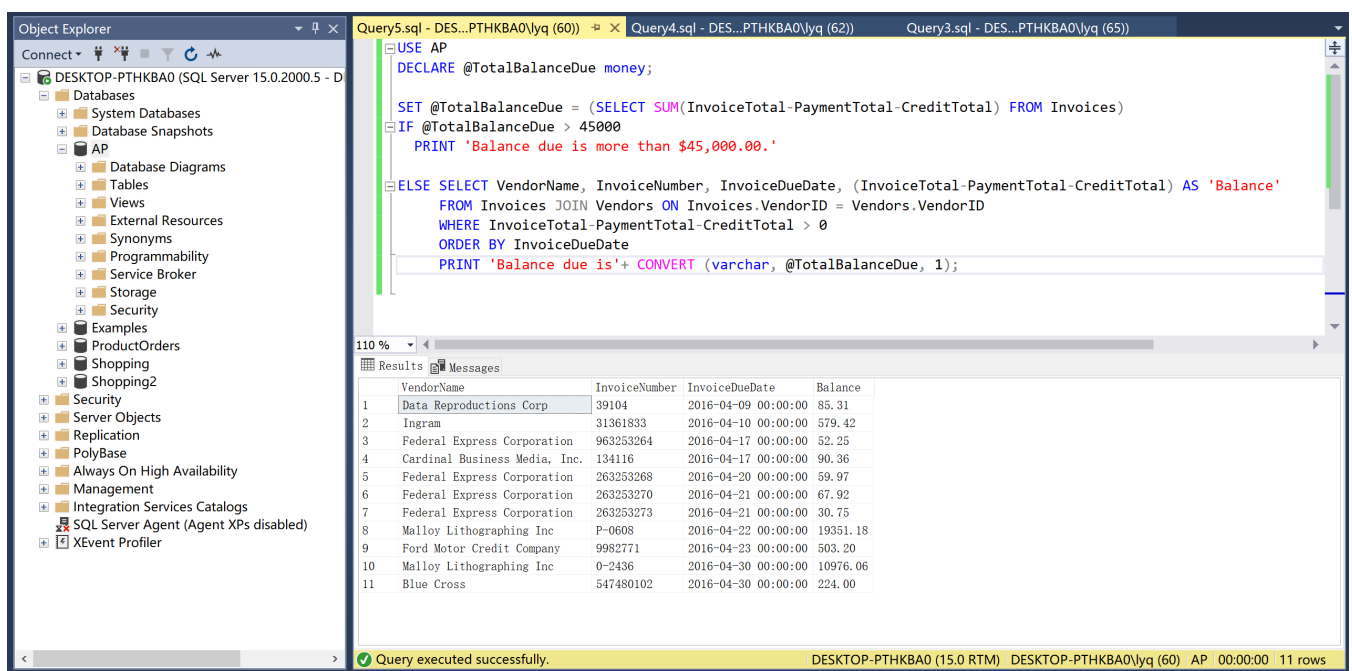
```
USE AP
SELECT *
FROM sys.foreign_keys
```

The results pane displays 6 rows of data, each representing a foreign key constraint. The columns include name, object_id, principal_id, schema_id, parent_object_id, type, type_desc, create_date, and modify_date.

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date
1	FK_InvoiceLineItems_GLAccounts	837578022	NULL	1	645577338	F	FOREIGN_KEY_CONSTRAINT	2020-09-05 16:14:01.147	2020-09-05
2	FK_InvoiceLineItems_Invoices	853578079	NULL	1	645577338	F	FOREIGN_KEY_CONSTRAINT	2020-09-05 16:14:01.150	2020-09-05
3	FK_Invoices_Terms	869578136	NULL	1	677577452	F	FOREIGN_KEY_CONSTRAINT	2020-09-05 16:14:01.150	2020-09-05
4	FK_Invoices_Vendors	885578193	NULL	1	677577452	F	FOREIGN_KEY_CONSTRAINT	2020-09-05 16:14:01.150	2020-09-05
5	FK_Vendors_GLAccounts	901578250	NULL	1	741577680	F	FOREIGN_KEY_CONSTRAINT	2020-09-05 16:14:01.153	2020-09-05
6	FK_Vendors_Terms	917578307	NULL	1	741577680	F	FOREIGN_KEY_CONSTRAINT	2020-09-05 16:14:01.153	2020-09-05

The status bar at the bottom indicates that the query was executed successfully, returning 6 rows in 00:00:00 seconds.

5. Write a script that declares and sets a variable named `@TotalBalanceDue`, which is equal to the total outstanding balance due. What is the datatype of the variable `@TotalBalanceDue`? If that balance due is less than \$45,000.00, the script should return a result set consisting of VendorName, InvoiceNumber, InvoiceDueDate, and Balance for each invoice with a balance due, sorted with the newest due date last. Then also return the value of `@TotalBalanceDue` in the format of “Balance due is ...”. If the total outstanding balance due is more than \$45,000.00, the script should return the message “Balance due is more than \$45,000.00”.



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server structure for 'DESKTOP-PTHKBA0 (SQL Server 15.0.2000.5 - D)'. The main pane shows a query script in 'Query5.sql'. The script declares a variable `@TotalBalanceDue` of type `money`, sets it to the sum of `InvoiceTotal - PaymentTotal - CreditTotal` from the `Invoices` table, and then uses an `IF` statement to either print a message or return a result set of invoices with a balance due greater than 0, ordered by `InvoiceDueDate`. The results pane shows 11 rows of data.

```
USE AP
DECLARE @TotalBalanceDue money;

SET @TotalBalanceDue = (SELECT SUM(InvoiceTotal-PaymentTotal-CreditTotal) FROM Invoices)
IF @TotalBalanceDue > 45000
    PRINT 'Balance due is more than $45,000.00.'
ELSE SELECT VendorName, InvoiceNumber, InvoiceDueDate, (InvoiceTotal-PaymentTotal-CreditTotal) AS 'Balance'
FROM Invoices JOIN Vendors ON Invoices.VendorID = Vendors.VendorID
WHERE InvoiceTotal-PaymentTotal-CreditTotal > 0
ORDER BY InvoiceDueDate
PRINT 'Balance due is'+ CONVERT (varchar, @TotalBalanceDue, 1);
```

	VendorName	InvoiceNumber	InvoiceDueDate	Balance
1	Data Reproductions Corp	39104	2016-04-09 00:00:00	85.31
2	Ingram	31361833	2016-04-10 00:00:00	579.42
3	Federal Express Corporation	963253264	2016-04-17 00:00:00	52.25
4	Cardinal Business Media, Inc.	134116	2016-04-17 00:00:00	90.36
5	Federal Express Corporation	263253268	2016-04-20 00:00:00	59.97
6	Federal Express Corporation	263253270	2016-04-21 00:00:00	67.92
7	Federal Express Corporation	263253273	2016-04-21 00:00:00	30.75
8	Malloy Lithographing Inc	P-0608	2016-04-22 00:00:00	19351.18
9	Ford Motor Credit Company	9982771	2016-04-23 00:00:00	503.20
10	Malloy Lithographing Inc	0-2436	2016-04-30 00:00:00	10976.06
11	Blue Cross	547480102	2016-04-30 00:00:00	224.00

Query executed successfully. DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (60) AP 00:00:00 11 rows

6. Explain the execution result generated by the following script. Then Write a script that generates the same result set but uses a temporary table in place of the derived table. Make sure your script tests for the existence of any objects it creates.

USE AP;

SELECT VendorName, LastInvoiceDate, InvoiceTotal

FROM Invoices

JOIN (SELECT VendorID, MAX(InvoiceDate) AS LastInvoiceDate
FROM Invoices

GROUP BY VendorID) AS LastInvoice

ON (Invoices.VendorID = LastInvoice.VendorID AND

Invoices.InvoiceDate = LastInvoice.LastInvoiceDate) JOIN Vendors ON

Invoices.VendorID = Vendors.VendorID ORDER BY VendorName,

LastInvoiceDate;

This script returns each vendor's the last invoice date and invoice total.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'DESKTOP-PTHKBA0'. The main window shows a query script in the 'Query6.sql' editor. The script consists of two parts: first, it creates a temporary table '#LastInvoice' by selecting the maximum invoice date for each vendor from the 'Invoices' table. Second, it joins this temporary table back to the 'Invoices' table and the 'Vendors' table to retrieve the vendor name, the last invoice date, and the invoice total, ordered by vendor name.

The results pane shows the output of the query, displaying 14 rows of data. The columns are VendorName, LastInvoiceDate, and InvoiceTotal.

VendorName	LastInvoiceDate	InvoiceTotal
Abbey Office Furnishings	2016-03-05 00:00:00	17.50
Bertelsmann Industry Svcs. Inc	2016-02-18 00:00:00	6940.25
Blue Cross	2016-04-01 00:00:00	224.00
Cahners Publishing Company	2016-02-28 00:00:00	2184.50
Cardinal Business Media, Inc.	2016-03-28 00:00:00	90.36
Coffee Break Service	2016-02-24 00:00:00	41.80
Compuserve	2016-01-13 00:00:00	9.95
Computerworld	2016-02-11 00:00:00	2433.00
Data Reproductions Corp	2016-03-10 00:00:00	85.31
Dean Witter Reynolds	2016-02-11 00:00:00	1367.50
Digital Dreamworks	2016-01-21 00:00:00	7125.34
Dristas Groom & McCormick	2016-01-23 00:00:00	220.00
Edward Data Services	2016-01-15 00:00:00	207.78
Evans Executone Inc	2015-12-24 00:00:00	95.00

The status bar at the bottom indicates that the query was executed successfully, returning 34 rows of data.

7. Write a script that generates the date and invoice total of the earliest invoice issued by each vendor, using a view instead of a derived table. Also write the script that creates the view, then use SELECT statement to show result of the view. Make sure that your script tests for the existence of the view. The view doesn't need to be redefined each time the script is executed.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server structure for 'DESKTOP-PTHKBA0 (SQL Server 15.0.2000.5 - D...)'.

The central pane shows a query script in 'Query7.sql - DES...PTHKBA0\lyq (53)'. The script includes the following T-SQL code:

```
USE AP
IF OBJECT_ID('EarliestInvoice') IS NOT NULL
    DROP VIEW EarliestInvoice;
IF OBJECT_ID('invoicetotal') IS NOT NULL
    DROP VIEW invoicetotal;

CREATE VIEW EarliestInvoice AS
SELECT VendorID, MIN(InvoiceDate) AS EarliestDate
FROM Invoices
GROUP BY VendorID;

CREATE VIEW invoicetotal AS
SELECT VendorName, EarliestDate, InvoiceTotal
FROM Invoices JOIN EarliestInvoice ON (Invoices.VendorID = EarliestInvoice.VendorID
AND Invoices.InvoiceDate = EarliestInvoice.EarliestDate)
JOIN Vendors ON Vendors.VendorID = EarliestInvoice.VendorID;

SELECT * FROM invoicetotal
```

The bottom pane shows the 'Results' tab with a table containing 14 rows of data. The status bar at the bottom indicates 'Query executed successfully.' and '34 rows'.

	VendorName	EarliestDate	InvoiceTotal
1	IBM	2016-01-07 00:00:00	116.54
2	Blue Cross	2016-02-03 00:00:00	224.00
3	Fresno County Tax Collector	2016-01-03 00:00:00	856.92
4	Data Reproductions Corp	2016-02-01 00:00:00	21842.00
5	Cardinal Business Media, Inc.	2016-02-22 00:00:00	175.00
6	Wang Laboratories, Inc.	2016-02-21 00:00:00	936.93
7	Reiter's Scientific & Pro Books	2016-03-19 00:00:00	600.00
8	Ingram	2016-02-03 00:00:00	1575.00
9	Computerworld	2016-02-11 00:00:00	2433.00
10	Edward Data Services	2016-01-15 00:00:00	207.78
11	Evans Executone Inc	2015-12-24 00:00:00	95.00
12	Wakefield Co	2016-03-20 00:00:00	356.48
13	Abbey Office Furnishings	2016-03-05 00:00:00	17.50
14	Pacific Bell	2015-12-30 00:00:00	16.33

8. Write a script that uses dynamic SQL to return a single column that represents the number of rows in the first table in the current database. The script should automatically choose the table that appears first alphabetically, and it should exclude tables named dtproperties and sysdiagrams. Name the column CountOfTable, where Table is the chosen table name. Show results for AP database.

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the 'DESKTOP-PTHKBA0' server with the 'AP' database selected. The main window shows a SQL query script in 'Query8.sql'. The script uses dynamic SQL to find the first table alphabetically in the 'AP' database, excluding 'dtproperties' and 'sysdiagrams', and returns the row count as 'CountOfTable'.

```
USE AP
DECLARE @TableName VARCHAR(120)
DECLARE @DynamicSQL VARCHAR(8000)

SET @TableName = (SELECT TOP 1 sys.tables.name
                  FROM sys.tables WHERE sys.tables.name NOT IN ('dtproperties', 'sysdiagrams')
                  ORDER BY sys.tables.name)

SET @DynamicSQL = 'SELECT COUNT(*) AS CountOfTable FROM ' + @TableName + ';'
PRINT @TableName;
EXEC(@DynamicSQL)
```

The Results pane shows a single row with the value 8 for the column CountOfTable.

	CountOfTable
1	8

The status bar at the bottom indicates the query was executed successfully on the 'AP' database, returning 1 row.