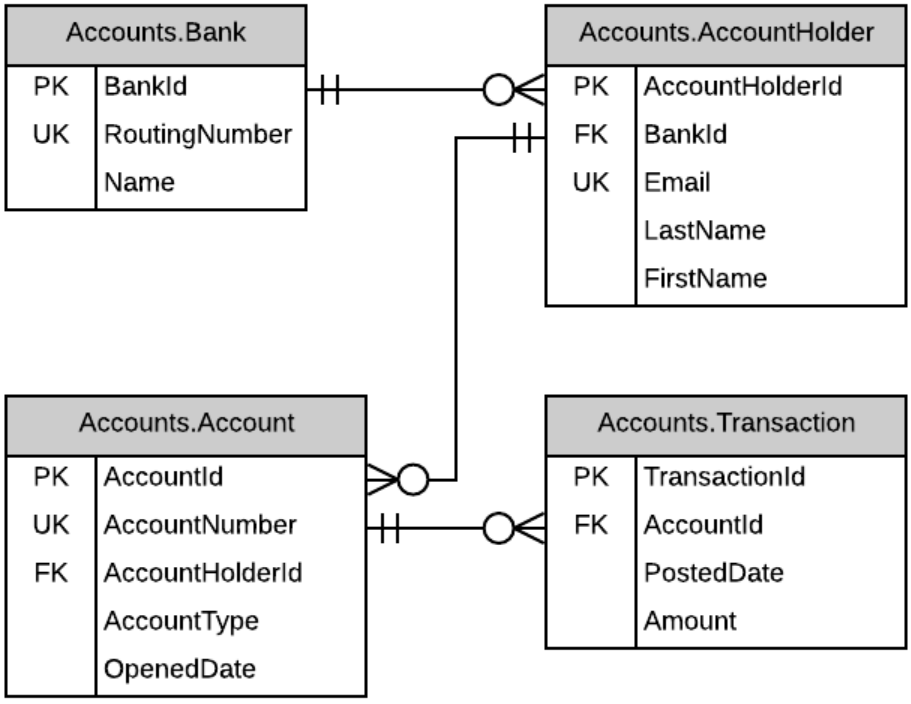
Exam 1 – Practice B Key

Database Schema

Use the below diagram as a reference to answer the questions on the following pages. For the most part, the complete column specifications are unimportant for the purposes of this exam. For all questions you may assume no column allows nulls.

NOTE: You may tear off this sheet for easier reference. You need not return this sheet.



Question 1 – **30 points** – Write a query that will return all account holders with total balances of at least $1,000,000. The total balance of an account holder is the sum of all transaction amounts on all his or her accounts.

For this problem, note that the column AccountType on Accounts.Account is of data type NCHAR(1).

Required Result Columns

AccountHolderId – The AccountHolderID from Accounts.AccountHolder.
LastName – The last name as it appears in LastName of Accounts.AccountHolder.
FirstName – The first name as it appears in FirstName of Accounts.AccountHolder.
CheckingBalance – The sum of Amount in Accounts.Transaction for all accounts of an account holder where AccountType is "C".
SavingsBalance – The sum of Amount in Accounts.Transaction for all accounts of an account holder where AccountType is "S".
TotalBalance – The sum of Amount in Accounts.Transaction for all accounts of an account holder, regardless of the account type.

Implementation Requirements

Your solution should contain no subqueries or window functions. The results should be sorted by TotalBalance in descending order, then by AccountHolderId in ascending order.

```
SELECT AH.AccountHolderId, AH.LastName, AH.FirstName,
   SUM(IIF(A.AccountType = N'C', T.Amount, 0.00)) AS CheckingBalance,
   SUM(IIF(A.AccountType = N'S', T.Amount, 0.00)) AS SavingsBalance,
   SUM(T.Amount) AS TotalBalance
FROM Accounts.AccountHolder AH
   INNER JOIN Accounts.Account A ON A.AccountHolderId = AH.AccountHolderId
   INNER JOIN Accounts.[Transaction] T ON T.AccountId = A.AccountId
GROUP BY AH.AccountHolderId, AH.LastName, AH.FirstName
HAVING SUM(T.Amount) >= 1000000
ORDER BY TotalBalance DESC, AH.AccountHolderId ASC;
```

Question 2 – **25 points** – Write a query that will return all accounts, along with their current balances and YTD activity, for the account holder having the email "john.doe@jmail.com". We want all accounts included, even if there is no activity on a given account.

For this problem, note that the column `PostedDate` on `Accounts.Transaction` is of data type `DATE`.

Required Result Columns

`AccountNumber` – The `AccountNumber` of `Accounts.Account`.
`AccountType` – The `AccountType` of `Accounts.Account`.
`YtdActivity` –The sum of `Amount` in `Accounts.Transaction` for all transactions posted on or after January 1, 2019. If an account has had no activity since January 1, 2019, the value should be 0.00.
`CurrentBalance` – The sum of `Amount` in `Accounts.Transaction` for all transactions posted on the account. If an account has had no activity, the value should be 0.00.

Implementation Requirements

Your solution should contain no subqueries or window functions. The results should be sorted by `AccountType` in ascending order, then by `AccountNumber` in ascending order.

```
SELECT A.AccountNumber, A.AccountType,
   SUM(IIF(T.PostedDate >= '2019-01-01', T.Amount, 0.00)) AS YtdActivity,
   ISNULL(SUM(T.Amount), 0.00) AS CurrentBalance
FROM Accounts.AccountHolder AH
   INNER JOIN Accounts.Account A ON A.AccountHolderId = AH.AccountHolderId
   LEFT JOIN Accounts.[Transaction] T ON T.AccountId = A.AccountId
WHERE AH.Email = N'john.doe@jmail.com'
GROUP BY A.AccountNumber, A.AccountType
ORDER BY A.AccountType ASC, A.AccountNumber ASC;
```

Question 3 – **25 points** – Write a query that will return all accounts, along with their 2019 opening balance and YTD activity, for the account holder having the email "john.doe@jmail.com". We want all accounts included, even if there is no activity on a given account.

For this problem, note that the column `PostedDate` on `Accounts.Transaction` is of data type `DATE`.

Required Result Columns

`AccountNumber` – The `AccountNumber` of `Accounts.Account`.
`AccountType` – The `AccountType` of `Accounts.Account`.
`OpeningBalance` – The sum of `Amount` in `Accounts.Transaction` for all transactions posted before January 1, 2019. If an account had no activity before January 1, 2019, the value should be 0.00.
`YtdActivity` –The sum of `Amount` in `Accounts.Transaction` for all transactions posted on or after January 1, 2019. If an account has had no activity since January 1, 2019, the value should be 0.00.

Implementation Requirements

Your solution should use two scalar-valued subqueries; one to calculate `OpeningBalance` and the other to calculate `YtdActivity`. The results should be sorted by `AccountType` in ascending order, then by `AccountNumber` in ascending order.

```
SELECT A.AccountNumber, A.AccountType,
   (
      SELECT ISNULL(SUM(T.Amount), 0.00)
      FROM Accounts.[Transaction] T
      WHERE T.AccountId = A.AccountId
         AND T.PostedDate < '2019-01-01'
   ) AS OpeningBalance,
   (
      SELECT ISNULL(SUM(T.Amount), 0.00)
      FROM Accounts.[Transaction] T
      WHERE T.AccountId = A.AccountId
         AND T.PostedDate >= '2019-01-01'
   ) AS YtdActivity
FROM Accounts.AccountHolder AH
   INNER JOIN Accounts.Account A ON A.AccountHolderId = AH.AccountHolderId
WHERE AH.Email = N'john.doe@jmail.com'
ORDER BY A.AccountType ASC, A.AccountNumber ASC;
```

Question 4 – **20 points** – Write a query that will return account holders at the bank with routing number 123456789, who have had an account opened before January 1, 2015.

For this problem, note that the column OpenedDate on Accounts.Account is of data type DATE.

Required Result Columns

LastName – The last name as it appears in LastName of Accounts.AccountHolder.
FirstName – The first name as it appears in FirstName of Accounts.AccountHolder.
Email – The email address as it appears in Email of Accounts.AccountHolder.

Implementation Requirements

Your solution should use a subquery to determine whether an account holder has an account that was opened before January 1, 2015. The results should be sorted by LastName in ascending order, then by FirstName in ascending order.

```
/***************************
 * Correlated Subquery
 ***************************/

SELECT AH.LastName, AH.FirstName, AH.Email
FROM Accounts.Bank B
   INNER JOIN Accounts.AccountHolder AH ON AH.BankId = B.BankId
WHERE B.RoutingNumber = N'123456789'
   AND EXISTS
   (
      SELECT *
      FROM Accounts.Account A
      WHERE A.AccountHolderId = AH.AccountHolderId
         AND A.OpenedDate < '2015-01-01'
   )
ORDER BY AH.LastName ASC, AH.FirstName ASC;

/***************************
 * Self-contained Subquery
 ***************************/

SELECT AH.LastName, AH.FirstName, AH.Email
FROM Accounts.Bank B
   INNER JOIN Accounts.AccountHolder AH ON AH.BankId = B.BankId
WHERE B.RoutingNumber = N'123456789'
   AND AH.AccountHolderId IN
   (
      SELECT A.AccountHolderId
      FROM Accounts.Account A
      WHERE A.OpenedDate < '2015-01-01'
   )
ORDER BY AH.LastName ASC, AH.FirstName ASC;
```