Data Analyst Submission

The Data Analyst interview questions and my answers are below, along with an explanation for some of the problems. If you have any questions, concerns or would like to see more of my work, feel free to contact me.

# Questions

For questions 1-4, use the following data model

**Person**

|  |
| --- |
| **PersonID** INT IDENTITY(1,1) PRIMARY KEY |
| FirstName VARCHAR(100) |
| LastName VARCHAR(100) |

**Sport**

|  |
| --- |
| **SportID** INT IDENTITY(1,1) PRIMARY KEY |
| SportName VARCHAR(100) |
| SportType VARCHAR(100) |

**History**

|  |
| --- |
| **HistoryID** INT IDENTITY(1,1) PRIMARY KEY |
| PersonID INT |
| SportID INT |
| Timestamp DATETIME |
| SCORE FLOAT |

### Question 1

**Write a query to list all members (FirstName, LastName) who played at least 2 sports in year 2010.**

Assuming you are looking for people who played at least 2 different sports (i.e. baseball and soccer), the code can be:

SELECT Person.FirstName, Person.LastName FROM (SELECT DISTINCT Person.PersonID, Person.FirstName, Person.LastName, Sport.SportType FROM Sport INNER JOIN (Person INNER JOIN History ON Person.PersonID = History.PersonID) ON Sport.SportID = History.SportID WHERE (((YEAR([History.Timestamp]))=2010))) GROUP BY PersonID, Person.FirstName, Person.LastName HAVING COUNT(1) > 1

If you want to include people who played that same sport multiple times in 2010 (i.e. baseball in June and baseball in July), then this code works:

SELECT Person.FirstName, Person.LastName FROM History INNER JOIN Person ON History.PersonID = Person.PersonID WHERE YEAR(History.[Timestamp]) = 2010 GROUP BY Person.FirstName, Person.LastName HAVING COUNT(1) > 1

### Question 2

**Write a query to list all sport type (SportType) that no one has ever played.**

Under the assumption that you want all sport types listed including repeats, the query is:

SELECT SportType FROM Sport WHERE SportID NOT IN (SELECT Distinct SportID FROM History)

If you do not want repeats, then a “Distinct” is added before SportType, and the code is:

SELECT Distinct SportType FROM Sport WHERE SportID NOT IN (SELECT Distinct SportID FROM History)

Another alternative way to code it (including repeats) is:

SELECT SportType FROM Sport LEFT OUTER JOIN History ON History.SportID = Sport.SportID WHERE History.SportID IS NULL

For the last coding method, here you need to outer join tables Sport and History, since you want ALL the records from table Sport, no matter if Sport records have any related History records.  For those Sport records that don't have any related History records, we are going to have NULL values in History.SportID field.  That effectively means that the History table contains no records about a given sport record.  That's exactly what we filter this dataset by in WHERE clause: we only want to see the records that have History.SportID = NULL.

### Question 3

**Write a query to list ALL SportType's (not name) and the total number of times played (2 columns: SportType, TotalNumberOfTimesPlayed) include unplayed SportTypes. Order by most played first.**

Since, NULL values are counted as zeros, any sport that was not played will not show up in the History table and thus will have a History.SportID equal to NULL. Since LEFT OUTER JOIN was used all SportTypes in the Sport table will be included and if it did not show in the History table then the count would 0 for the NULL values.

SELECT Sport.SportType, COUNT(History.SportID) AS TotalNumberOfTimesPlayed FROM Sport LEFT OUTER JOIN History ON History.SportID = Sport.SportID GROUP BY Sport.SportType ORDER BY COUNT(History.SportID) DESC

### Question 4

**Write a query that returns a single row for each person and columns of the score for the date of each game. Example:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **First Name** | **Last Name** | **Sport Name** | **1/1/18** | **2/1/18** | **3/1/18** |
| Jane | Doe | Golf | 78 | 82 | 75 |

This problem requires using PIVOT or CROSSTAB to turn the date portion of the History.Timestamp field into column headers / field names. The below text is written in MS Access SQL.

TRANSFORM Sum(History.Score) AS SumOfScore SELECT Person.FirstName, Person.LastName, Sport.SportName FROM (History INNER JOIN Person

ON History.PersonID = Person.PersonID) INNER JOIN Sport ON History.SportID = Sport.SportID GROUP BY Person.FirstName, Person.LastName, Sport.SportName PIVOT Format(History.[Timestamp],"Short Date")

Helpful Link that I found: <https://www.red-gate.com/simple-talk/sql/t-sql-programming/creating-cross-tab-queries-and-pivot-tables-in-sql/>

### Question 5

Given the following table structure:

**dbo.limeadeAccounts**

|  |
| --- |
| LimeadeAccountId INT IDENTITY(1,1) PRIMARY KEY |
| EmployerID INT |
| Email VARCHAR(100) |
| Name VARCHAR(100) |

**dbo.Employer**

|  |
| --- |
| EmployerID INT IDENTITY(1,1) PRIMARY KEY |
| EmployerName VARCHAR(100) |
| ProgramName VARCHAR(100) |
| PrimaryColor1 VARCHAR(100) |
| PrimaryColor2 VARCHAR(100) |

**dbo.IncentiveLevelLog**

|  |
| --- |
| LimeadeAccountId INT |
| IncentiveLevel INT |
| DateCreated TIMESTAMP |
| DateEarned TIMESTAMP |
| IncentiveValue INT |

Write a query or queries that will produce two files – a detail record file that contains a row for the incentive value and a row for the imputed income (2/3rds of the incentive value) for each Level earned in the previous month, and a control and balance file that contains the record count, value summation, and run date of the detail file. Example output files (use for reference and output formatting): <https://limeade.brickftp.com/f/66f7f752fa0f50f8>

**Assumptions**:

* There are 3 possible incentive levels a user can earn
* A Level 1 incentive is worth $100
* A Level 2 incentive is worth $250
* A Level 3 incentive is worth $500

I used MySQL for this problem.

SELECT E.EmployerID, ILL.IncentiveValue

FROM dbo.Employer E

JOIN dbo.LimeadeAccounts LA

ON E.EmployerID = LA.EmployerID

JOIN dbo.IncentiveLevelLog ILL

ON LA.LimeadeAccountId = ILL.LimeadeAccountId

UNION ALL

SELECT E.EmployerID, IncentiveValue \* (2.0/3.0)

FROM dbo.Employer E

JOIN dbo.LimeadeAccounts LA

ON E.EmployerID = LA.EmployerID

JOIN dbo.IncentiveLevelLog ILL

ON LA.LimeadeAccountId = ILL.LimeadeAccountId;

INTO OUTFILE 'Result.csv';

SELECT (COUNT(E.EmployerID))\*2, SUM(IncentiveValue \* (5.0/3.0)), NOW()

FROM dbo.Employer E

JOIN dbo.LimeadeAccounts LA

ON E.EmployerID = LA.EmployerID

JOIN dbo.IncentiveLevelLog ILL

ON LA.LimeadeAccountId = ILL.LimeadeAccountId;

INTO OUTFILE 'CB.csv';