**UWS Course COMP09003**

**Business Intelligence**

**CNET Data Mart**

**Stuart Hood**

**December 2011**

Contents

[Introduction to the business and website. 2](#_Toc311639696)

[General Business Overview 2](#_Toc311639697)

[Website analysis 4](#_Toc311639698)

[Interaction with users and data capture. 4](#_Toc311639699)

[Internal users 6](#_Toc311639700)

[Analytical requirements 7](#_Toc311639701)

[Data Mart design for single business process 8](#_Toc311639702)

[Dimensional Model 8](#_Toc311639703)

[Data Dictionary 10](#_Toc311639704)

[Data Mart implementation 13](#_Toc311639705)

[Database Diagram 13](#_Toc311639706)

[Queries and results with BI value 14](#_Toc311639707)

[Appendix 20](#_Toc311639708)

[Appendix 1: OLTP database diagram 20](#_Toc311639709)

[Appendix 2: Fact table extract SQL 21](#_Toc311639710)

[Appendix 3: Fact table ETL 22](#_Toc311639711)

# Introduction to the business and website.

## General Business Overview

CNET.com (now a subsidiary of CBS Corp which reportedly acquired the company for $1.8 billion) is an IT media website specializing in reviews of tech consumer electronics and related products. The website is sub-divided into five sections dedicated to Reviews, News, Download, CNET TV and How To articles. The award-winning News website has been a pioneer in the field of technology news reporting since 1996. CNET TV offers online video content of tech product reviews and “first-looks.” The ‘How To’ site was launched in August 2011 and is dedicated to providing tutorials, guides and tech tips to end users.

The largest part of the site is dedicated to Reviews of all manner of technology products and offers product ratings with Editor’s Choice Awards for the most innovative or functional releases.

The Download section, which will be the focus of this work, offers over 60,000 software applications, PC starter kits, games, security patches and more for Windows and Macintosh desktop and mobile devices and, according to cbsinteractive.com, provides over 2.5 million downloads per day on a ‘Try Before You Buy’ basis.



The original software publishers include many blue-chip companies but the site will accept uploads from any source and, after scanning for any embedded malware, will make any application (with the usual ‘appropriateness’ caveats) freely available to consumers. The vast bulk of amateur and small-business releases are entirely freeware.

As for those applications which have a price, the ‘Free to Try’ policy and the fact that users then pay the original software publisher if they decide to keep the software, means that CNET Downloads is almost entirely advertising driven, though it’s entirely possible there’s some commission arrangement involved where they’re hosting such high-value products as Microsoft Office.

The exception to the above is where the website has a promotional offer as will be discussed later.

CBS claim the following at <http://www.cbsinteractive.com/advertise/media-kit/download.html>

|  |
| --- |
| **Advertiser Benefits** |
| * Established, trusted and safe environment to distribute software
* Reaching an active audience seeking software solutions—one in three users download new software, applications and tools each day
* Communicating with an audience who influence at least six other people about brands and products
* Being a part of an immersive environment to develop a meaningful dialogue with consumers on their terms, through online, video, podcasts and integrated sponsorships
 |

|  |
| --- |
| **Audience Profile** |
| * Unique Visitors Per Month: 18.5 million
* Page Views Per Month: 209 million
* Time Spent on CNET Per Month1: 7:39 minutes
* Average Age: 39 years old
* Male/Female: 79%/21%
* Average Income: $84,356
* College Educated: 85%
* Home/Work/School: 95% broadband
* Average Next 12 Months HW/CE: $2,206 plan to spend
 |

Publishers, advertisers and audience will all be discussed in the section that follows.

# Website analysis

## Interaction with users and data capture.

The only time a visitor needs to create an account and log in is when they are taking advantage of a promotion and have to provide CNET with bank account details. Otherwise it’s entirely possible to visit the site and download applications without ever explicitly giving any personal details.

However, even at first visit, unless the user has gone to some lengths their web browser will already have informed the server of not only its own name and version, but also the name and version of its host operating system so e.g. Windows users will find that option pre-selected in the ‘Find Software’ area – see home page image above.

Whether a user creates an account or not, unless they have disabled cookies the site recognizes them and if they’ve previously viewed any products or created any lists these will be available.

|  |  |  |
| --- | --- | --- |
|  |  |  |

However, the site’s ability to track users goes far beyond this, as declare in the following screen shot from their privacy policy pages.



Perhaps more revealing is the following taken from the same source as the previous screen shot.

**CBS - Third-Party Online Advertising**

Some of the advertisements you see on CBS Interactive Web Sites are delivered by third parties who also collect information through cookies, web beacons, and other technologies about your online activities, either on our Web Sites or across the Internet, in an effort to understand your interests and deliver you advertisements that are tailored to your interests. These third parties include advertisers, advertising agencies, and ad networks that may collect information when you view or interact with one of their advertisements.

Third parties that currently collect information on our Sites for the purpose of serving you with advertising tailored to your interest include, but are not limited to, Atlas, BlueKai, DoubleClick, EyeWonder, Google, Mediamind, Audience Science and Pointroll.

A quick look at the html source for a page will reveal that not only do they host their own advertisements, but that some of the truly dynamic ads are Flash (or other javascript-related) animations that may forward the user to another site but are also capable of providing CNET and/or the advert provider and/or their ‘Referrer’ partners with a host of information about who clicked what, when and where they’d previously been.

In conclusion then, the website takes full advantage of the panoply of ‘clickstream’ technologies to garner information about its visitors and, vitally, to bill advertisers on a ‘click-through’ basis. Add this to the ubiquitous providers of demographic data and CNET and its partners, advertisers and software publishers have potential access to details of even ‘anonymous’ visitors.

As to users who explicitly create an account with the website, besides the standard account creation form CNET offer a quick sign up facility using the user’s Facebook, Google or Yahoo account as seen in the following screen shot .



Obviously users who create a site account using a social networking login are providing even more demographic and other information that will be of enormous use to the site in targeting advertising, hence the site’s use of reduced-price promotions to induce users to create an account.

Once an account has been created, the user can take advantage of these promotional offers after providing even more personal information in the form of banking details.



## Internal users

Within CNET there are disparate groups that will want access to all the visitor data being collected. From a business perspective, perhaps the most important of these will be those concerned with quantifying the site’s effectiveness at promoting sales of publisher’s products, discovering which producers and categories of product “sell” (or don’t) to which demographics and targeting advertising appropriately and especially to promoting ‘high-value’ publishers and products.

For this reason, the data mart I propose will be centred on the following areas:

* Analysing visitor demographics and relating them to applications downloaded
* Relating download rates to the viewing of reviews and adverts
* Analysing payment methods and values of products actually purchased from the site
* Studying the performance of different software publishers
* Studying the performance of different advert providers

To this end I’ll ignore how the demographic data is gathered and simply start from the fact that there’s a wealth of such data in the system. Similarly, I’ll assume that systems are in place to track and record the various click-stream events and concentrate on those that will provide best business intelligence in the context of the above user requirements.

# Analytical requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Visitor | Application | Download | Publisher | Advert | Advert Provider | Review |
| Track user actions | ✓ | ✓ | ✓ | ✓ | ✓ |  | ✓ |
| Provide apps, ads & reviews |  | ✓ |  | ✓ | ✓ | ✓ | ✓ |
| Register users | ✓ |  |  |  |  |  |  |

To provide the business intelligence required to fulfil the information requirements of the business users, dimensions for analysing the website processes will involve those in the above matrix. Given a data mart with such a structure, general queries to provide the required information could fall into the following groups

* Queries to relate visitor demographics to the applications they download and the adverts they have seen
* Queries to relate applications downloaded to the reviews that visitors have read
* Queries to analyse promotional downloads
* Queries to analyse the onward value to the various application publishers
* Queries to compare the effectiveness of adverts from the various providers used

Ideally, the queries would return low-level fine-grained details which could then be aggregated to provide the user with a higher level overview allowing the user to drill down into and “slice and dice” as required, though summarizing queries more suitable for reports are obviously also highly useful.

# Data Mart design for single business process

## Dimensional Model

My primary consideration when it came to design was how to relate user downloads to their previous history on the site as regards the adverts they might have seen and the reviews they may have read.

On the live CNET website, the web application itself will obviously have its own OLTP database to provide the on-screen content and persist many of the user’s interactions, but the clickstream data and recording of other interactions with the site will be coming from the web servers’ own transaction logs, the shared session state server’s dedicated database and the wealth of data provided by the partner companies providing the click-through and referrer information. Presumably there is an ETL process that gathers all this low-level server and third-party data together so it can be related to the CNET OLTP data before final ETL to the data warehouse.

To best try and mimic this process the OLTP database I designed revolved around two tables, one to record all session data and another for all the events. Though not a requirement for this project I’ve included the database diagram as an appendix for reference. The Session table holds nothing more than the VisitorID and the start and end date times of the session. In reality, most sessions will end after the default expiry set on the server unless the user has signed in and explicitly signs out. The event table holds details of each event recorded which, in this context, is the viewing of an advert, the reading of a review or the download of an application and includes details of how (if) the user was logged in at the time, since even known users with an account can visit and download without logging in. The other tables then mimic the actual OLTP database in a standard normalized way.

For the data mart, I then combined the session and event tables to form the fact table, with each row relating to a single event and the SessionID present as a degenerate dimension that can be tied together through the VisitorID and/or the AppID. Since I’ve no way of knowing how CNET actually make money at the ‘per-click’ level the primary measure will be a row count, i.e. the combined number of specific events, but I’ve included the list price of the app involved as I’m sure CNET will want to know the onward value to their customers (the application publishers) of the downloads they are hosting, and the promo price of downloads where CNET directly receive payment from the visitor.

I also de-normalized the transaction tables, especially regarding the geography details of visitors, publishers and advert providers as these will no doubt be of interest to a company that has region-specific portals for Australia, China, France, Germany, Japan, South-East Asia, Turkey, Taiwan, and the UK.

As per recommended practice, to avoid null keys in the fact table a value of 1 for DownloadID, AdvertID and ReviewID relates to a Type of ‘NONE’ in the dimensions.

As a final note, although visual studio data generator created data for 2000 events, to get sessions to overlap in time I had to compress them all into a limited period – January 2011 - which doesn’t allow for any meaningful queries to compare downloads over separate time periods. Similarly, I’ve considered any considerations of different time zones my fictional users may be in as beyond the scope of this project though it’s obviously something CNET will consider.

Conceptual Dimensional Model



## Data Dictionary

**Fact Session Event**

|  |  |  |  |
| --- | --- | --- | --- |
| **DM Column Name** | **Datatype** | **Description** | **Values** |
| SessionID | int, PK1, NN | Degenerate Dimension |  |
| VisitorID | Int, PK1, FK, NN | Surrogate |  |
| LoginTypeID | int, PK1, FK, NN | Surrogate |  |
| SessionStartDate | int, PK1, FK, NN | Surrogate | ETL: CONVERT(INT,CONVERT(VARCHAR(8),s.start,112)) |
| SessionEndDate | int, PK1, FK, NN | Surrogate | ETL: CONVERT(INT,CONVERT(VARCHAR(8),s.[End],112)) |
| SessionStartTime | int, PK1, FK, NN | Surrogate | ETL: CONVERT(CHAR(8),s.Start,108) |
| SessionEndTime | int, PK1, FK, NN | Surrogate | ETL: CONVERT(CHAR(8),s.[End], 108) |
| EventStartDate | int, PK1, FK, NN | Surrogate | ETL: CONVERT(INT,CONVERT(VARCHAR(8),e.EventStartDateTime,112)) |
| EventEndDate | int, PK1, FK, NN | Surrogate | ETL: CONVERT(INT,CONVERT(VARCHAR(8),e.EventEndDateTime,112)) |
| EventStartTime | int, PK1, FK, NN | Surrogate | ETL: CONVERT(CHAR(8),e.EventStartDateTime,108) |
| EventEndTime | int, PK1, FK, NN | Surrogate | ETL: CONVERT(CHAR(8),e.EventEndDateTime,108) |
| AppID | int, PK1, FK, NN | Surrogate |  |
| AdvertID | int, PK1, FK, NN | Surrogate |  |
| DownloadID | int, PK1, FK, NN | Surrogate |  |
| ReviewID | int, PK1, FK, NN | Surrogate |  |
| Session Duration | intComputed Column | Measure | SessionEndTime-SessionStartTime \* |
| List Value | money | Measure |  |
| Promo Value | money | Measure |  |

\* The following would have been much preferable and runs in Management Studio with a SELECT sub-query on the date/time fields but fails as a computed column. Still working on it ☹

DATEDIFF(ss,

DATEADD(ss,SessionStartTime,

CAST(CAST(SessionStartDate AS VARCHAR(8)) AS smalldatetime)),

DATEADD(ss, SessionEndTime,

CAST(CAST(SessionEndDate AS CHAR(8)) AS smalldatetime)))

**Dim Visitor**

|  |  |  |  |
| --- | --- | --- | --- |
| VisitorID | Int, PK, Identity, NN |  |  |
| Visitor Alt Key | int, NN |  |  |
| RegistrationType | nvarchar(10) | How (if) registered | NONE, CNET, FACEBOOK, GOOGLE, YAHOO |
| RegistrationDate | date, not null |  |  |
| Visitor Region | nvarchar(10) | Continental Region | AMER, EMEA, APAC |
| Visitor Country | nvarchar(20) | Name of country | ~10 per region |
| Visitor Town | nvarchar(20) | Name of town | ~10 per country |
| Age | nvarchar(10) | Age group | 10-1920-2930-3940-4950-5960-6970+UNKNOWN |
| Ethnicity | nvarchar(10) | Ethnic background | CAUC, AFRAM,ASIAN,HISP, OTHER, UNKNOWN |
| Gender | nvarchar(10) | Visitor gender | MALE, FEMALE, UNKNOWN |
| Parent  | nvarchar(10) | Is visitor a parent? | YES, NO, UNKNOWN |
| Income | nvarchar(20) | Income range | 0-24,99925,000-49,99950,000-99,999100,000+UNKNOWN |
| Education | nvarchar(10) | Visitor’s educational background | SCHOOL, COLLEGE, POSTGRAD, UNKNOWN |

**Dim Advert**

|  |  |  |  |
| --- | --- | --- | --- |
| AdvertID | int, PK, Identity, NN |  |  |
| Advert Alt Key | int |  |  |
| AdvertType | nvarchar(10) | The source of advert | NONE, CNET, EXT (EXTERNAL) |
| Provider Name | nvarchar(20) | Advert Provider | Provider of external ad |
| Provider Region | nvarchar(10) | Continental Region | AMER, EMEA, APAC |
| Provider Country | nvarchar(20) | Name of country | ~10 per region |
| Provider Town | nvarchar(20) | Name of town | Data Generator random string |
| Advert Start Date | date |  |  |
| Advert End Date | date |  |  |

**Dim Application**

|  |  |  |  |
| --- | --- | --- | --- |
| AppID | int, PK, Identity, NN |  |  |
| App Alt Key | Natural key |  |  |
| App Name | nvarchar(20) | Product Name | Data Generator random string |
| App Version | nvarchar(20) | Product Version | Home, Pro, Premium |
| App Category | nvarchar(20) | Product Category | SECURITY, HOME, INTERNET, EDUCATIONAL, BUSINESS |
| License Type | nvarchar(20) | Freeware or ‘Try & buy’ | FREE, TRIAL |
| User Rating | tinyint | Customer input | 1 – 5 |
| Publisher | nvarchar(20) | Software publisher | Data Generator random string |
| Publisher Region | nvarchar(10) | Continental Region | AMER, EMEA, APAC |
| Publisher Country | nvarchar(20) | Name of country | ~10 per region |
| Publisher Town | nvarchar(20) | Name of town | Data Generator random string |
| Price | money | List price of application | Data Generator £1-500 |
| Date App Added | date |  |  |
| Date App Removed | date |  |  |

**Dim Review**

|  |  |  |  |
| --- | --- | --- | --- |
| ReviewID | int, PK, Identity, NN |  |  |
| Review Alt Key | Source Key |  |  |
| Review Type | nvarchar(10) |  | NONE, CNET |
| Reviewer | nvarchar(20) | Name of CNET reviewer | Data Generator random string |
| Review Date | date |  |  |
| Review Rating | tinyint | Reviewer rating of app | Data Generator random 1-5 |

**Dim Download**

|  |  |  |  |
| --- | --- | --- | --- |
| DownloadID | int, PK, Identity, NN |  |  |
| Download Alt Key | int |  |  |
| Download Type | nvarchar(10) |  | NONE, CNET |
| Payment Type | nvarchar(10) | Denormalized | FREEWARE, TRIAL, MASTERCARD, VISA, DISCOVER, AMEX, PAYPAL |
| Promo Type | nvarchar(10) | CNET or Publisher | NONE, CNET, VENDOR |

**Dim Login Type**

|  |  |  |  |
| --- | --- | --- | --- |
| LoginTypeID | int, PK, Identity, NN |  |  |
| Login Type Alt Key | int |  |  |
| Login Type | nvarchar(10) | Login status during event | ANON, KNOWN,CNET |

# Data Mart implementation

## Database Diagram

There’s little more to be added to what has previously been said regarding design. To get the data from the OLTP database to the data mart I used SSIS. The SQL to extract the fact table data is in the appendix with a screen shot of the lookup transformations. The date values were converted into the Dim Date primary key integer format during extraction and injected directly into the fact table along with the measures.



## Queries and results with BI value

The first thing to look at is the number of downloads where the visitor has, during some session, clicked on an advert for the application that has been downloaded and view this across a selection of visitor demographic groupings. This would give the business some idea of the groups most likely to download an application after having read an advert and thus target future advertising.

--Visitor demographics for downloads in January 2011

--after visitor clicks ad during any session

SELECT v.[Visitor Gender],v.[Visitor Region],v.[Visitor Education], v.[Visitor Age], v.[Visitor Income], COUNT(f.DownloadID) AS Downloads

FROM [Fact Session Event] f, [Dim Visitor] v, [Dim Date] d

WHERE f.VisitorID=v.VisitorID

AND f.SessionStartDate=d.ID

AND d.[Year]='2011' AND d.[MonthName]='January'

AND f.DownloadID>1

AND f.VisitorID IN (

SELECT VisitorID FROM [Fact Session Event] WHERE AdvertID>1)

GROUP BY GROUPING SETS(

(v.[Visitor Gender], v.[Visitor Region], v.[Visitor Education], v.[Visitor Age], v.[Visitor Income]),())



Secondly, a similar query but rolled up across various attributes of the applications downloaded after a visitor has read a review. This could let the business know where to concentrate their product review activities.

--Applications downloaded after visitor reads the review during any session

SELECT app.AppID, app.[App Category],app.[License Type],app.[App Version], app.[User Rating], COUNT(f.downloadid) AS Downloads

FROM [Fact Session Event] f, [Dim Application] app, [Dim Visitor] v

WHERE f.VisitorID=v.VisitorID

AND f.AppID=app.AppID

AND f.DownloadID>1

AND f.VisitorID IN (

SELECT VisitorID FROM [Fact Session Event] WHERE ReviewID>1)

GROUP BY GROUPING SETS(ROLLUP(app.AppID, app.[App Category],app.[License Type],app.[App Version],app.Publisher,app.[User Rating]))



This next query outputs an analysis cube which looks at payments actually received by CNET for applications that have been on promotional offer. This type of information would be useful when considering whether to allow users to register using other social networking sites or to pay by methods other than the traditional credit/debit card.

--Payment Type & value by Visitor.[Registration Type] by App for promo payments received.

SELECT f.AppID, v.[Registration Type] AS [Visitor Registration],d.[Payment Type],

SUM(f.[Promo Value]) AS Income

FROM [Fact Session Event] f, [Dim Download] d, [Dim Visitor] v

WHERE f.DownloadID=d.DownloadID

AND f.VisitorID=v.VisitorID

AND f.DownloadID>1

AND d.[Promo Type] IS NOT NULL

GROUP BY GROUPING SETS(CUBE(f.AppID, v.[Registration Type], f.VisitorID,d.[Payment Type]))



Now we look at the onward value to the customers (software publishers) as the list price of the applications that are being downloaded on a ‘Try-before-you-buy’ basis. This is exactly the sort of information required when it comes to deciding pricing scales when these publishers then want to advertise their products on the CNET site.

--List Value of downloads by app publisher

SELECT f.AppID, app.Publisher, f.[List Value],

SUM([List Value]) OVER(PARTITION BY [Publisher]) AS [Total Value],

COUNT([List Value]) OVER(PARTITION BY [publisher]) AS [Total Downloads]

FROM [Fact Session Event] f, [Dim Application] app

WHERE f.AppID=app.AppID

AND f.[List Value] IS NOT NULL

AND DownloadID>1



This query is aimed at ranking the effectiveness of the various advertising providers used by CNET and their customers, again by looking at the list values of applications downloaded. Obviously useful information when it comes to deciding who to employ for creating adverts and how much they can fairly expect to be paid for their services.

--Summed List Value of downloads Ranked by Ad Agency

SELECT f.AppID, a.[Provider Name] AS [Advert Provider], SUM(f.[List Value]) AS [SUM List Value],

RANK() OVER(PARTITION BY [Provider Name] ORDER BY SUM([List Value])) AS [RANK],

DENSE\_RANK() OVER(PARTITION BY [Provider Name] ORDER BY SUM([List Value])) AS [DENSE\_RANK]

FROM [Fact Session Event] f, [Dim Advert] a

WHERE f.AppID=a.[Application ID]

AND f.DownloadID>1

GROUP BY f.AppID, a.[Provider Name]



The previous queries all having been at quite a fine granular level, this final one generates an overall report according to two of the attributes CNET can be most certain of the validity of, namely how their visitors have registered and the categories of applications that those visitor groups download. This would obviously be extremely useful in determining which users new applications in a particular category should be targeted towards.

--Dynamic Pivot for all downloads by Visitor Registration Type by Application Category

DECLARE @RegType VARCHAR(200)

SELECT @RegType = STUFF((SELECT DISTINCT '],[' + [Registration Type] FROM [Dim Visitor]

FOR XML PATH('')),1,2,'') + ']'

DECLARE @AppCategory VARCHAR(200)

SELECT @AppCategory = STUFF((SELECT DISTINCT ''',''' + [App Category] FROM [Dim Application]

FOR XML PATH('')),1,2,'') + ''''

DECLARE @query VARCHAR(500)

SET @query = 'SELECT \* FROM (SELECT a.[App Category], v.[Registration Type] AS RegType, f.DownloadID AS DownloadID

FROM [Fact Session Event] f, [Dim Visitor] v, [Dim Application] a

WHERE f.VisitorID=v.VisitorID

AND f.AppID=a.AppID

AND f.DownloadID>1 AND [App Category] IN ('+@AppCategory+')) AS src

PIVOT(COUNT(DownloadID) FOR RegType IN ('+@RegType+')) AS pvt'

EXEC(@query)



As would be expected with a functional data mart there are many other queries that could be run but the forgoing give a good overall picture of what is possible with even such a simplistic design and implementation and effectively answer the type of questions raised during the analysis phase.

# Appendix

## Appendix 1: OLTP database diagram



## Appendix 2: Fact table extract SQL

SELECT e.EventID, s.SessionID, v.VisitortId, l.TypeID AS LoginTypeID, app.appid, ad.AdID, d.DOWNLOADID, r.REVID, app.Price AS ListValue, PROMOPRICE = NULL,

CONVERT(INT,CONVERT(VARCHAR(8),s.start,112)) AS SessionStartDate,

CONVERT(INT,CONVERT(VARCHAR(8),s.[End],112)) AS SessionEndDate,

CONVERT(CHAR(8),s.Start,108) AS SessionStartTime,

CONVERT(CHAR(8),s.[End], 108) AS SessionEndTime,

CONVERT(INT,CONVERT(VARCHAR(8),e.EventStartDateTime,112))AS EventStartDate,

CONVERT(INT,CONVERT(VARCHAR(8),e.EventEndDateTime,112)) AS EventEndDate,

CONVERT(CHAR(8),e.EventStartDateTime,108) AS EventStartTime,

CONVERT(CHAR(8),e.EventEndDateTime,108) AS EventEndTime,

FROM [Event] e, [SESSION] s, Visitor v, LoginType l, Advert ad, [APPLICATION] app, REVIEW r, DOWNLOAD d

WHERE e.SessionID=s.SessionID

AND S.VisitorID=v.VisitortId

AND e.LoginTypeID=l.TypeID

AND e.AdID=ad.AdID

AND e.DownloadID=d.DOWNLOADID

AND e.ReviewID=r.REVID

AND ad.AppId=app.appid

UNION ALL

SELECT e.EventID, s.SessionID, v.VisitortId, l.TypeID AS LoginTypeID, app.appid, ad.AdID, d.DOWNLOADID, r.REVID, app.Price AS ListValue, p.PROMOPRICE,

CONVERT(INT,CONVERT(VARCHAR(8),s.start,112)) AS SessionStartDate,

CONVERT(INT,CONVERT(VARCHAR(8),s.[End],112)) AS SessionEndDate,

CONVERT(CHAR(8),s.Start,108) AS SessionStartTime,

CONVERT(CHAR(8),s.[End], 108) AS SessionEndTime,

CONVERT(INT,CONVERT(VARCHAR(8),e.EventStartDateTime,112))AS EventStartDate,

CONVERT(INT,CONVERT(VARCHAR(8),e.EventEndDateTime,112)) AS EventEndDate,

CONVERT(CHAR(8),e.EventStartDateTime,108) AS EventStartTime,

CONVERT(CHAR(8),e.EventEndDateTime,108) AS EventEndTime,

FROM [Event] e

JOIN [SESSION] s ON (e.SessionID=s.SessionID)

JOIN LoginType l ON (e.LoginTypeID=l.TypeID)

JOIN Advert ad ON (e.AdID=ad.AdID)

JOIN DOWNLOAD d ON (e.DownloadID=d.DOWNLOADID)

JOIN REVIEW r ON (e.ReviewID=r.REVID)

JOIN Visitor v ON (v.VisitortId=s.visitorid)

LEFT OUTER JOIN PROMOTION p ON (d.PROMOID=p.PROMOID)

JOIN [APPLICATION] app ON (d.APPID=app.AppId)

UNION ALL

SELECT e.EventID, s.SessionID, v.VisitortId, l.TypeID AS LoginTypeID, app.appid, ad.AdID, d.DOWNLOADID, r.REVID, app.Price AS ListValue, PROMOPRICE = NULL ,

CONVERT(INT,CONVERT(VARCHAR(8),s.start,112)) AS SessionStartDate,

CONVERT(INT,CONVERT(VARCHAR(8),s.[End],112)) AS SessionEndDate,

CONVERT(CHAR(8),s.Start,108) AS SessionStartTime,

CONVERT(CHAR(8),s.[End], 108) AS SessionEndTime,

CONVERT(INT,CONVERT(VARCHAR(8),e.EventStartDateTime,112))AS EventStartDate,

CONVERT(INT,CONVERT(VARCHAR(8),e.EventEndDateTime,112)) AS EventEndDate,

CONVERT(CHAR(8),e.EventStartDateTime,108) AS EventStartTime,

CONVERT(CHAR(8),e.EventEndDateTime,108) AS EventEndTime,

FROM [Event] e, [SESSION] s, Visitor v, LoginType l, Advert ad,[APPLICATION] app, REVIEW r, DOWNLOAD d

WHERE e.SessionID=s.SessionID

AND S.VisitorID=v.VisitortId

AND e.LoginTypeID=l.TypeID

AND e.AdID=ad.AdID

AND e.DownloadID=d.DOWNLOADID

AND e.ReviewID=r.REVID

## Appendix 3: Fact table ETL

