Business Intelligence Sample Project:

The following report is concerned with: the design and implementation of an efficient, scalable OLTP database responsible for the record of application usage transactions, the design and implementation of an efficient and scalable OLAP data warehouse responsible for generating valuable usage metrics, and the respective python scripting of the ETL process that bridges the two.

Parker Ferguson

12/19/2013

Executive Summary

"AwesomeApp offers two packages to its users: the notSoAwesome plan and the moreThanAwesome plan. The information stored for each user includes his/her email address, age, and gender. The notSoAwesome plan allows a user to take advantage of only the basic features. The moreThanAwesome plan offers additional features for team collaboration. For this advanced user plan, team sizes can vary (no limit on team size imposed). However, each team has to be owned by a particular user and has a name assigned to it by the owner. A user can be the owner as well as a regular member of multiple teams."

The 'notSoAwesome' plan allows a user to take advantage of only the basic features. The 'moreThanAwesome' plan offers additional features for team collaboration

Deliverables

a) Design a data model to store the data efficiently (in a relational database like MySQL)

b) Determine what metrics we need to identify user trends and design the data model to make metrics calculation easy and fast

c) Develop a rudimentary Extract-Transform-Load (ETL) procedure that will do the mapping from the original table(s) in the database to derived data (to be used in metrics calculation)

Satisfaction criteria

- ✓ Sketch of the ERD diagrams for the data model
- ✓ Code for the ETL procedure(s), preferably in Shell Script, Perl, or Python. However, pseudo code would also be acceptable, provided you maintain good coding practice.

Coding requirements

- ✓ Clean code
- ✓ Succinct yet clear documentation and commenting
- ✓ Error handling
- ✓ Well abstracted
- ✓ Unit tested

Assumptions

- Both the "moreThanAwesome" plan and the "notSoAwesome" plan offer its users a tool for collaboration; this is assumed to mean that both provide the capability for users to collaborate on projects and to do so via teams

- The defining difference between the two plans is that the "moreThanAwesome" plan allows for unlimited team size, whereas the "notSoAwesome" plan allows for team participation, but limits the number of members

- Team size is determined by the owners plan status; a team's size limit is set by the plan on which the owner is registered, regardless of the plan type any one member of this team may have; handled via database TRIGGER "maxSizeLimit"

- As user age is a changing metric, age will be substituted by date of birth "user_dateBirth"

- Users must access projects via a team; users must be members of a team to access any project. A single user team is the default at sign up; Teams are "owned" by a single user, team owner is assumed the authority to add members. Each project is "worked_on" by a single team.

- Team owner must be a member of all teams he/she owns; handled via database TRIGGER "ownerMustBeOnTeam"

- User cancellation date, or whether a user is active or not (cancelled) is considered beyond the scope of this project; see "Recommendations"

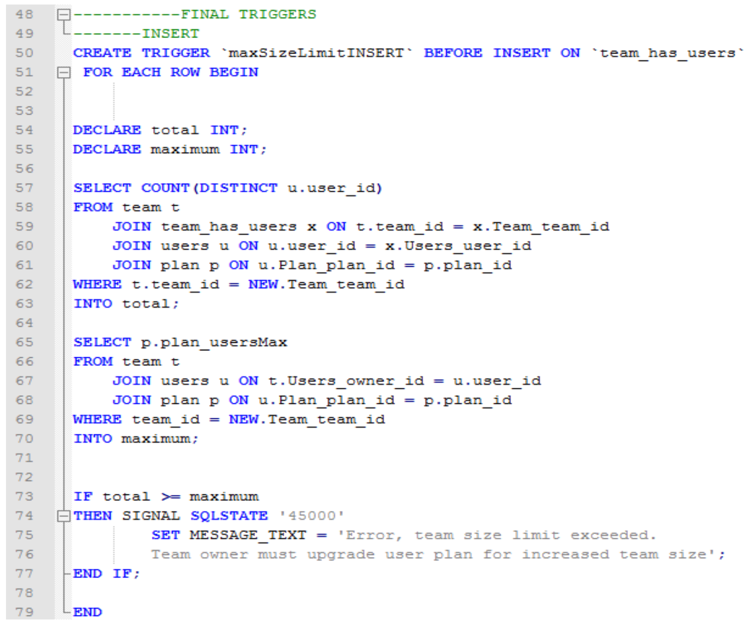
- MySQLdb Driver is assumed to be on users machine for import ; executables of both 32 bit and 64 bit are included; see folder "mySQLdb_Executables_32&64"

- We are able to generate common data from our application including:
    - Ip address & location
        - Location will be assumed to be retrieved by application and not database
    - Session login/logout DATETIME
    - User creation DATE
    - Team creation DATE
    - Project creation and completion DATETIME
        - User may be prompted for completion verification as indicated by significant time since last activity (see "considerations")
        - Login dashboard design may also be an avenue to indicate completion; see "Considerations"

INCLUDED

OLTP Database:

- Trigger "maxSizeLimit": restricts INSERT and UPDATE on "Team_has_Users" table; limit is determined by owners plan type and will not allow a new team member addition if team size limit will be exceeded (trigger included in folder“dbSQLSnippets)



```
-----------FINAL TRIGGERS
-------INSERT
CREATE TRIGGER `maxSizeLimitINSERT` BEFORE INSERT ON `team_has_users`
 FOR EACH ROW BEGIN


DECLARE total INT;
DECLARE maximum INT;

SELECT COUNT(DISTINCT u.user_id)
FROM team t
    JOIN team_has_users x ON t.team_id = x.Team_team_id
    JOIN users u ON u.user_id = x.Users_user_id
    JOIN plan p ON u.Plan_plan_id = p.plan_id
WHERE t.team_id = NEW.Team_team_id
INTO total;

SELECT p.plan_usersMax
FROM team t
    JOIN users u ON t.Users_owner_id = u.user_id
    JOIN plan p ON u.Plan_plan_id = p.plan_id
WHERE team_id = NEW.Team_team_id
INTO maximum;


IF total >= maximum
THEN SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error, team size limit exceeded.
        Team owner must upgrade user plan for increased team size';
END IF;

END
```

- Trigger “userGender”: restricts INSERT or UPDATE of gender on “Users” table; requires ‘M’ or ‘F’ be entered and nothing else (trigger included in foler “dbSQLSnippets)

- Trigger “ownerMustBeOnTeam”: AFTER INSERT or UPDATE of team table, owner is added to team he/she owns; owner must be on team that owner “owns” (trigger included in folder “dbSQLSnippets)

Data Warehouse:

- Star schema
- Index on all primary keys as well as “users_plan_id” & “project_team_id” as compensation for collapse of tables Users JOIN Plan & Project JOIN Team

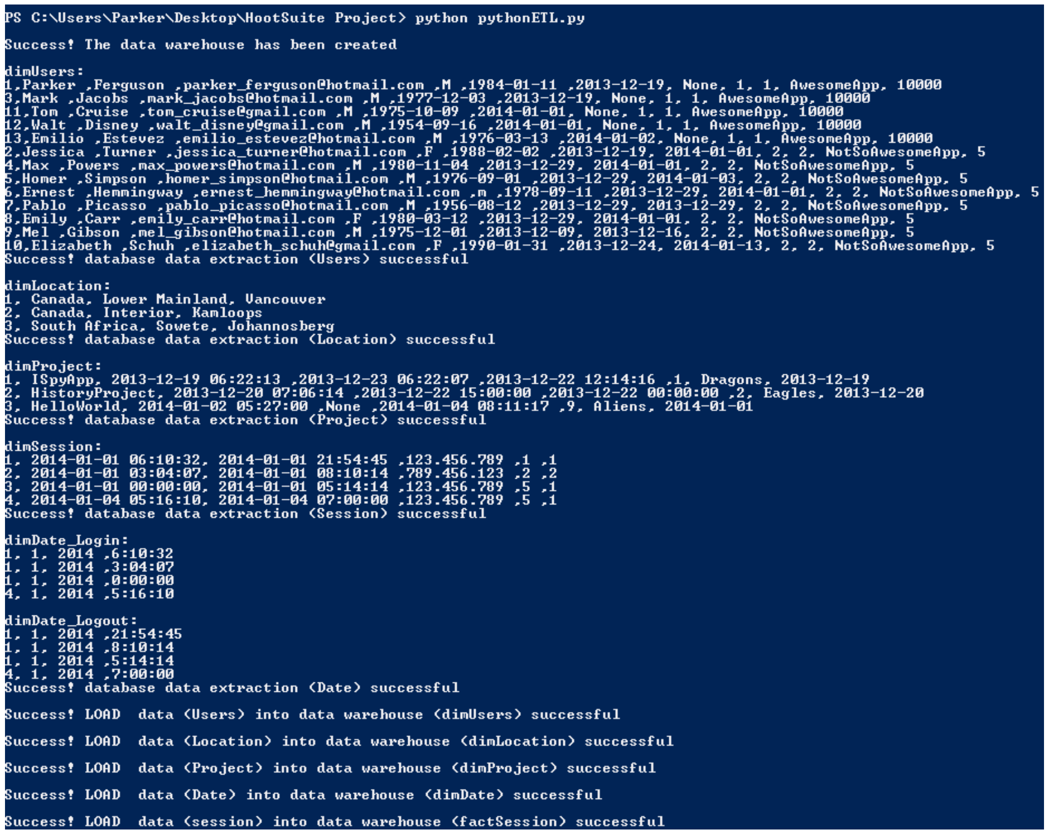
SQL Scripts:

- Sample SQL scripts for:
    - DATE to day, month, or year format testing for dimDate transformation
    - TRIGGER and trigger related snippets
    - Database build SQL
    - Data warehouse build SQL

ETL Scripting:

- TRANFORMATION implementation on Date to dimDate table; pull day, month, year, and time from DATETIME
- Data warehouse build, EXTRACT and LOAD of data:
    - Users and Plan to dimUsers
    - Project and Team to dimProject
    - Location to dimLocation

    - session_loginDate and session_logoutDate columns of session table to dimDate
- Selection of ONLY required data, exclusion of data not relevant to warehouse
- Printing of Extracted data
- Printing of success/failure of data extraction/load



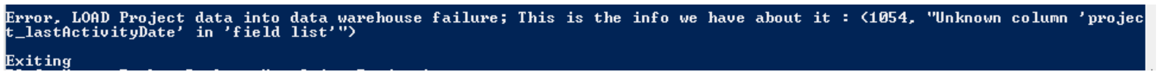
```
PS C:\Users\Parker\Desktop\HootSuite Project> python pythonETL.py

Success! The data warehouse has been created

dimUsers:
1,Parker ,Ferguson ,parker_ferguson@hotmail.com ,M ,1984-01-11 ,2013-12-19, None, 1, 1, AwesomeApp, 10000
3,Mark ,Jacobs ,mark_jacobs@hotmail.com ,M ,1977-12-03 ,2013-12-19, None, 1, 1, AwesomeApp, 10000
11,Tom ,Cruise ,tom_cruise@gmail.com ,M ,1975-10-09 ,2014-01-01, None, 1, 1, AwesomeApp, 10000
12,Walt ,Disney ,walt_disney@gmail.com ,M ,1954-09-16 ,2014-01-01, None, 1, 1, AwesomeApp, 10000
13,Emilio ,Estevez ,emilio_estevez@hotmail.com ,M ,1976-03-13 ,2014-01-02, None, 1, 1, AwesomeApp, 10000
2,Jessica ,Turner ,jessica_turner@hotmail.com ,F ,1988-02-02 ,2013-12-19, 2014-01-01, 2, 2, NotSoAwesomeApp, 5
4,Max ,Powers ,max_powers@hotmail.com ,M ,1980-11-04 ,2013-12-29, 2014-01-01, 2, 2, NotSoAwesomeApp, 5
5,Homer ,Simpson ,homer_simpson@hotmail.com ,M ,1976-09-01 ,2013-12-29, 2014-01-03, 2, 2, NotSoAwesomeApp, 5
6,Ernest ,Hemmingway ,ernest_hemmingway@hotmail.com ,m ,1978-09-11 ,2013-12-29, 2014-01-01, 2, 2, NotSoAwesomeApp, 5
7,Pablo ,Picasso ,pablo_picasso@hotmail.com ,M ,1956-08-12 ,2013-12-29, 2013-12-29, 2, 2, NotSoAwesomeApp, 5
8,Emily ,Carr ,emily_carr@hotmail.com ,F ,1980-03-12 ,2013-12-29, 2014-01-01, 2, 2, NotSoAwesomeApp, 5
9,Mel ,Gibson ,mel_gibson@hotmail.com ,M ,1975-12-01 ,2013-12-09, 2013-12-16, 2, 2, NotSoAwesomeApp, 5
10,Elizabeth ,Schuh ,elizabeth_schuh@gmail.com ,F ,1990-01-31 ,2013-12-24, 2014-01-13, 2, 2, NotSoAwesomeApp, 5
Success! database data extraction (Users) successful

dimLocation:
1, Canada, Lower Mainland, Vancouver
2, Canada, Interior, Kamloops
3, South Africa, Sowete, Johannosberg
Success! database data extraction (Location) successful

dimProject:
1, ISpyApp, 2013-12-19 06:22:13 ,2013-12-23 06:22:07 ,2013-12-22 12:14:16 ,1, Dragons, 2013-12-19
2, HistoryProject, 2013-12-20 07:06:14 ,2013-12-22 15:00:00 ,2013-12-22 00:00:00 ,2, Eagles, 2013-12-20
3, HelloWorld, 2014-01-02 05:27:00 ,None ,2014-01-04 08:11:17 ,9, Aliens, 2014-01-01
Success! database data extraction (Project) successful

dimSession:
1, 2014-01-01 06:10:32, 2014-01-01 21:54:45 ,123.456.789 ,1 ,1
2, 2014-01-01 03:04:07, 2014-01-01 08:10:14 ,789.456.123 ,2 ,2
3, 2014-01-01 00:00:00, 2014-01-01 05:14:14 ,123.456.789 ,5 ,1
4, 2014-01-04 05:16:10, 2014-01-04 07:00:00 ,123.456.789 ,5 ,1
Success! database data extraction (Session) successful

dimDate_Login:
1, 1, 2014 ,6:10:32
1, 1, 2014 ,3:04:07
1, 1, 2014 ,0:00:00
4, 1, 2014 ,5:16:10

dimDate_Logout:
1, 1, 2014 ,21:54:45
1, 1, 2014 ,8:10:14
1, 1, 2014 ,5:14:14
4, 1, 2014 ,7:00:00
Success! database data extraction (Date) successful

Success! LOAD  data (Users) into data warehouse (dimUsers) successful

Success! LOAD  data (Location) into data warehouse (dimLocation) successful

Success! LOAD  data (Project) into data warehouse (dimProject) successful

Success! LOAD  data (Date) into data warehouse (dimDate) successful

Success! LOAD  data (session) into data warehouse (factSession) successful
```

Unit Testing:

- Incremental physical testing; see "Diagrams"

Error Handling:

- Db: Team size violations handled via specific error message
- Db: Gender violations generate specific error message
- ETL: filterwarnings used to ignore "Can't DROP database, database does not exist" warning
- ETL: All procedures contained in Try/Catch blocks with specific error messaging combined with system warning (custom error handling)



```
Error, LOAD Project data into data warehouse failure; This is the info we have about it : (1054, "Unknown column 'projec
t_lastActivityDate' in 'field list'")

Exiting
```

Metrics

- Usage levels via session length
  - Usage by age
  - Usage by gender
  - Usage by location (ip)

    Indicative of need for application redesign
    Indicative of need for market targeting
- Number of new users per period (date)
  - By location, age, and gender

    Allows for market targeting
- Number of upgrades by period (date)
  - By location, age, and gender

    Allows for market targeting
- Teams close to exceeding size limits

    Allow for upgrade suggestion to team owner
- Ability to generate emails/contact info of users who:
  - Have not logged in in some period of time
  - Have not upgraded in some period of time
  - Have been a member for some period of time

    Allows for drip marketing campaign
- Project turnaround by plan type
  - Data to support benefits of upgrade

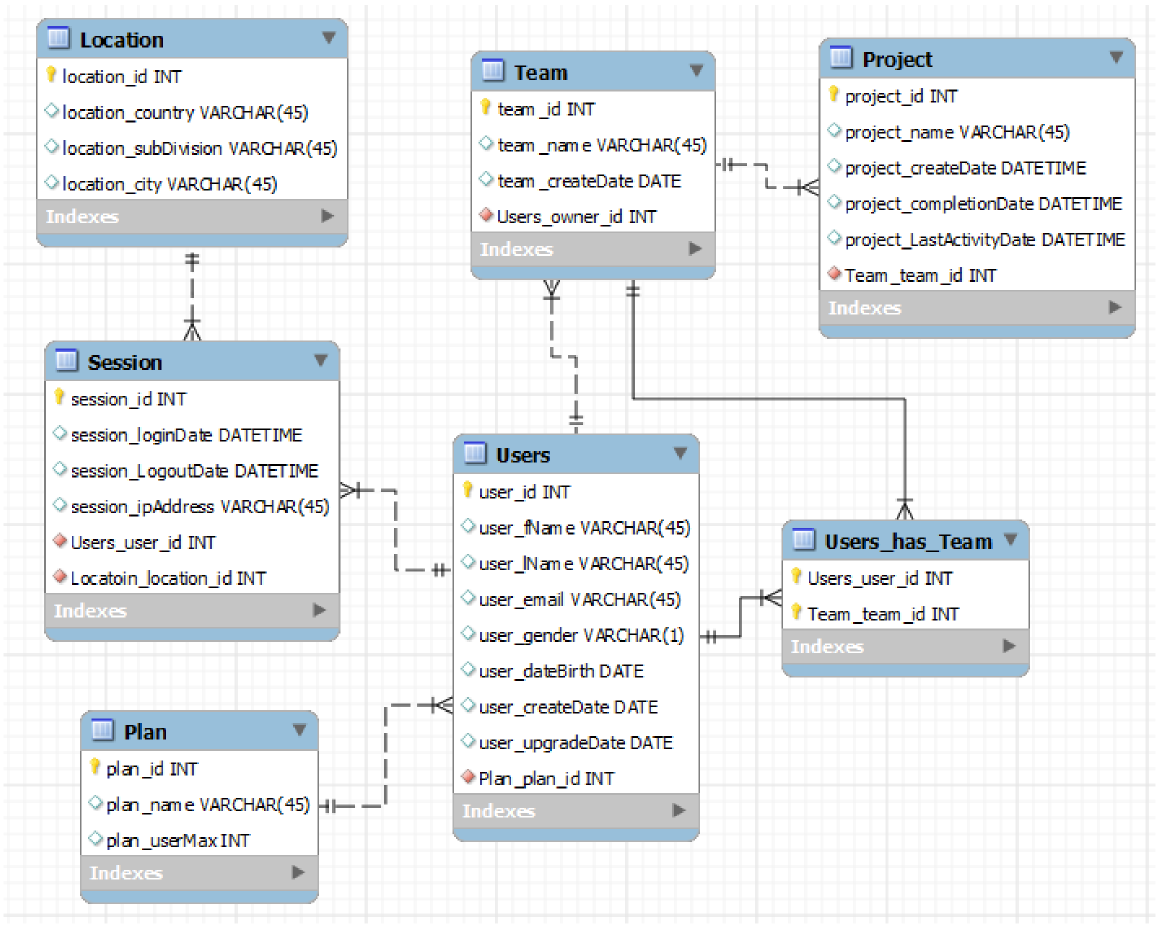
    Provides a selling point for user upgrade

Tools

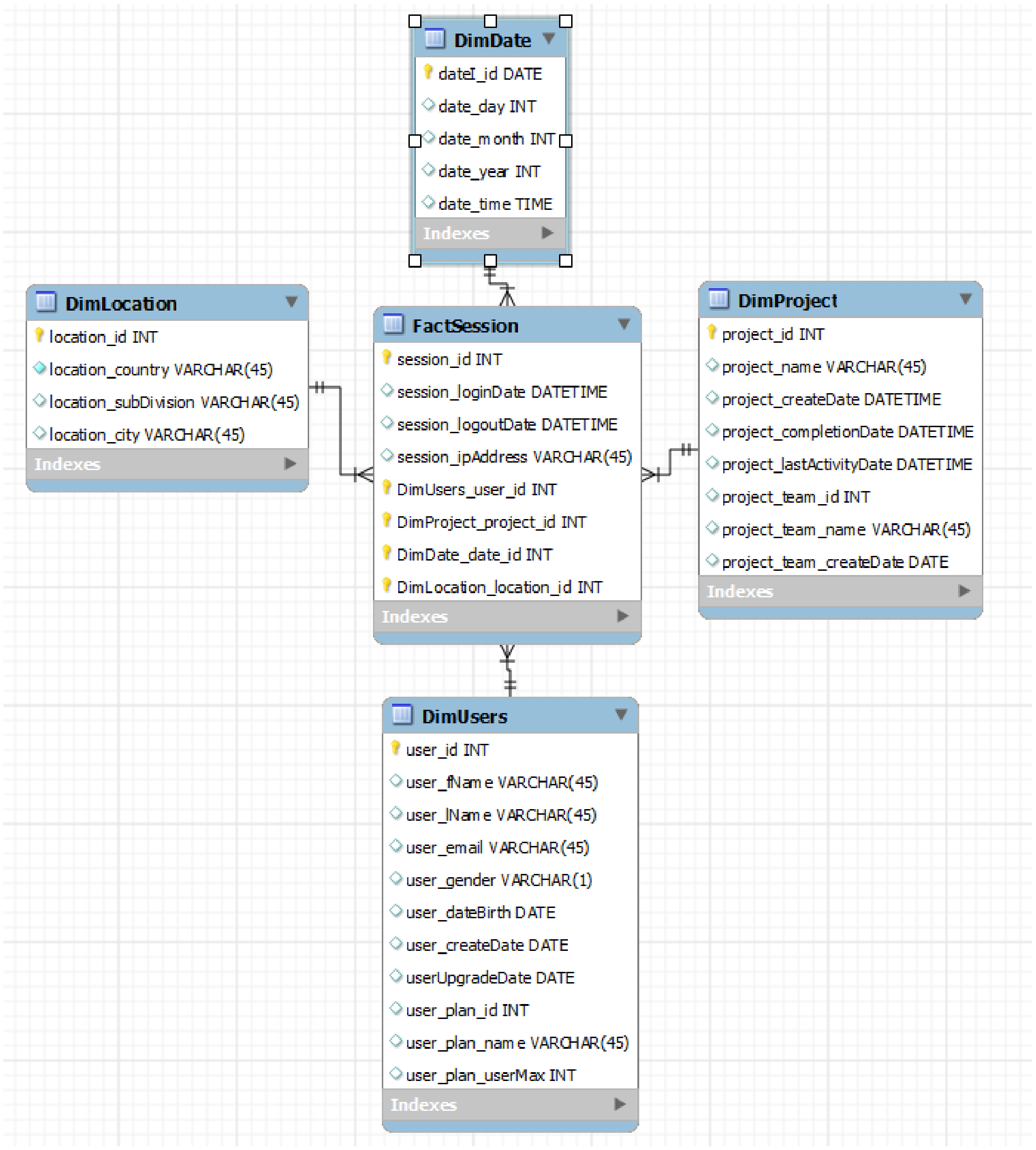
- MySql Dashboard: database design
- PhpMyAdmin: Database build
- Python: ETL scripting
  - Python MySqldb driver (included)
- Query Forge testing suite

DIAGRAMS

OLTP DATABASE



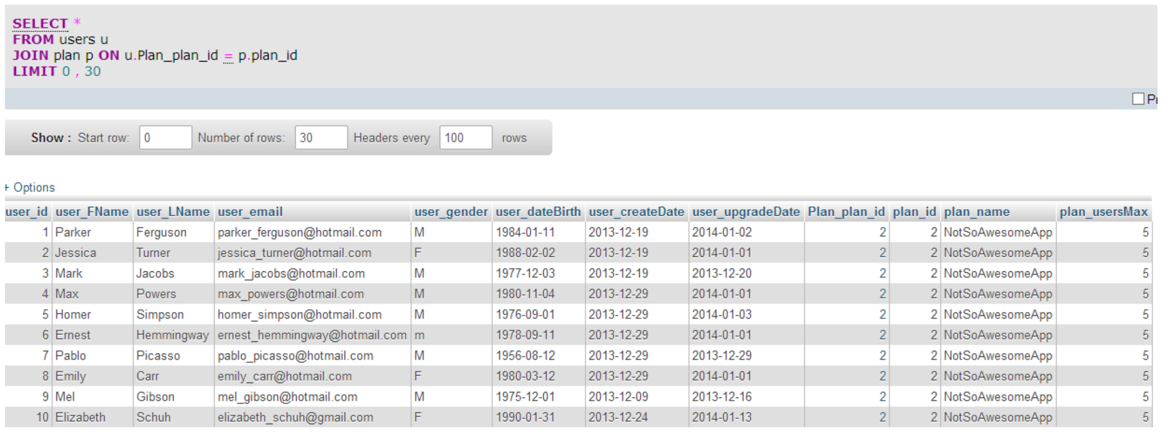
OLAP Data warehouse: Star Schema



UNIT TEST & ANALYSIS

Physical Test #1:

Database: "SELECT * FROM USERS u JOIN plan p ON u.Plan_plan_id = p.plan_id"



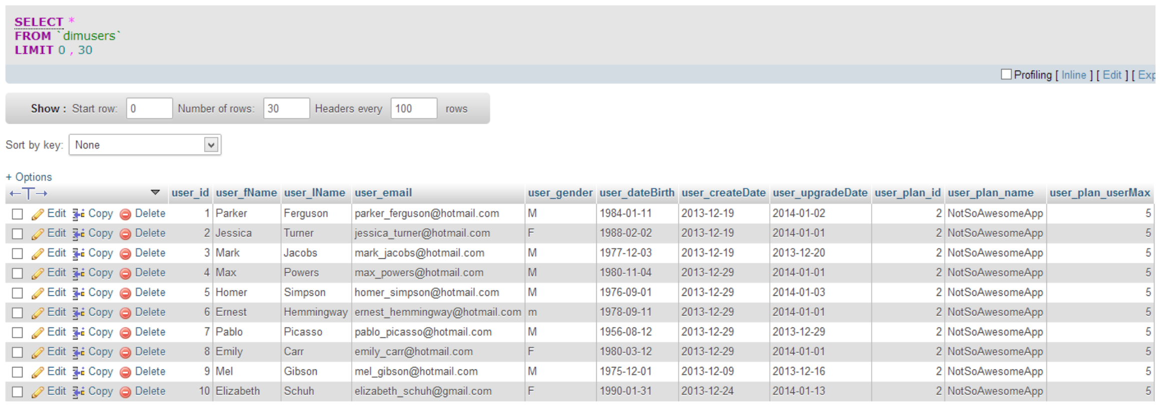
```
SELECT *
FROM users u
JOIN plan p ON u.Plan_plan_id = p.plan_id
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

user_id	user_FName	user_LName	user_email	user_gender	user_dateBirth	user_createDate	user_upgradeDate	Plan_plan_id	plan_id	plan_name	plan_usersMax
1	Parker	Ferguson	parker_ferguson@hotmail.com	M	1984-01-11	2013-12-19	2014-01-02	2	2	NotSoAwesomeApp	5
2	Jessica	Turner	jessica_turner@hotmail.com	F	1988-02-02	2013-12-19	2014-01-01	2	2	NotSoAwesomeApp	5
3	Mark	Jacobs	mark_jacobs@hotmail.com	M	1977-12-03	2013-12-19	2013-12-20	2	2	NotSoAwesomeApp	5
4	Max	Powers	max_powers@hotmail.com	M	1980-11-04	2013-12-29	2014-01-01	2	2	NotSoAwesomeApp	5
5	Homer	Simpson	homer_simpson@hotmail.com	M	1976-09-01	2013-12-29	2014-01-03	2	2	NotSoAwesomeApp	5
6	Ernest	Hemmingway	ernest_hemmingway@hotmail.com	m	1978-09-11	2013-12-29	2014-01-01	2	2	NotSoAwesomeApp	5
7	Pablo	Picasso	pablo_picasso@hotmail.com	M	1956-08-12	2013-12-29	2013-12-29	2	2	NotSoAwesomeApp	5
8	Emily	Carr	emily_carr@hotmail.com	F	1980-03-12	2013-12-29	2014-01-01	2	2	NotSoAwesomeApp	5
9	Mel	Gibson	mel_gibson@hotmail.com	M	1975-12-01	2013-12-09	2013-12-16	2	2	NotSoAwesomeApp	5
10	Elizabeth	Schuh	elizabeth_schuh@gmail.com	F	1990-01-31	2013-12-24	2014-01-13	2	2	NotSoAwesomeApp	5

Data warehouse: "SELECT * FROM dimUsers"



```
SELECT *
FROM `dimusers`
LIMIT 0 , 30
```

☐ Profiling [Inline] [Edit] [Exp

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

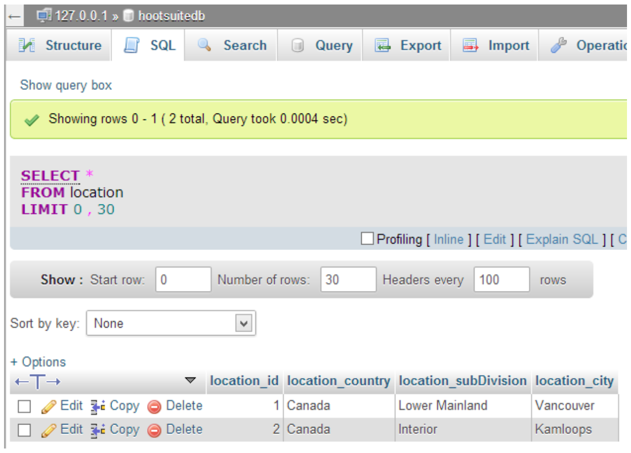
Sort by key: None

+ Options

←T→	user_id	user_fName	user_lName	user_email	user_gender	user_dateBirth	user_createDate	user_upgradeDate	user_plan_id	user_plan_name	user_plan_userMax
☐ Edit Copy Delete	1	Parker	Ferguson	parker_ferguson@hotmail.com	M	1984-01-11	2013-12-19	2014-01-02	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	2	Jessica	Turner	jessica_turner@hotmail.com	F	1988-02-02	2013-12-19	2014-01-01	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	3	Mark	Jacobs	mark_jacobs@hotmail.com	M	1977-12-03	2013-12-19	2013-12-20	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	4	Max	Powers	max_powers@hotmail.com	M	1980-11-04	2013-12-29	2014-01-01	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	5	Homer	Simpson	homer_simpson@hotmail.com	M	1976-09-01	2013-12-29	2014-01-03	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	6	Ernest	Hemmingway	ernest_hemmingway@hotmail.com	m	1978-09-11	2013-12-29	2014-01-01	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	7	Pablo	Picasso	pablo_picasso@hotmail.com	M	1956-08-12	2013-12-29	2013-12-29	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	8	Emily	Carr	emily_carr@hotmail.com	F	1980-03-12	2013-12-29	2014-01-01	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	9	Mel	Gibson	mel_gibson@hotmail.com	M	1975-12-01	2013-12-09	2013-12-16	2	NotSoAwesomeApp	5
☐ Edit Copy Delete	10	Elizabeth	Schuh	elizabeth_schuh@gmail.com	F	1990-01-31	2013-12-24	2014-01-13	2	NotSoAwesomeApp	5

Physical Test #2:

Database: “SELECT * FROM Location”



Showing rows 0 - 1 (2 total, Query took 0.0004 sec)

```
SELECT *
FROM location
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [C

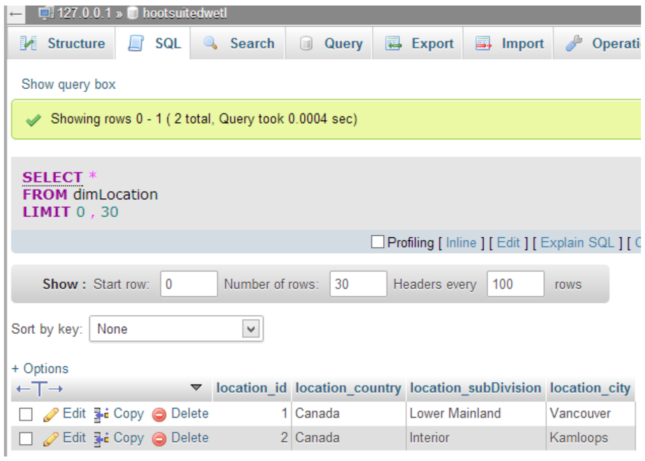
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	location_id	location_country	location_subDivision	location_city
Edit Copy Delete	1	Canada	Lower Mainland	Vancouver
Edit Copy Delete	2	Canada	Interior	Kamloops

Data warehouse: “SELECT * FROM dimLocation”



Showing rows 0 - 1 (2 total, Query took 0.0004 sec)

```
SELECT *
FROM dimLocation
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [C

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

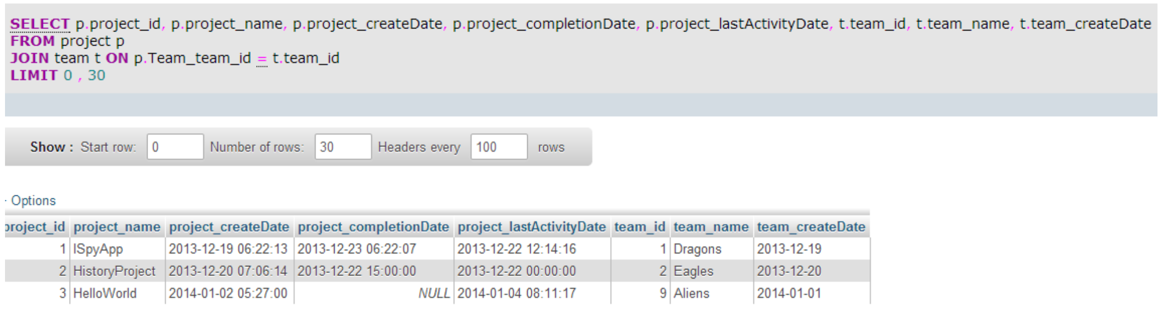
Sort by key: None

+ Options

	location_id	location_country	location_subDivision	location_city
Edit Copy Delete	1	Canada	Lower Mainland	Vancouver
Edit Copy Delete	2	Canada	Interior	Kamloops

Physical Test #3:

Database: “SELECT p.project_id, p.project_name, p.project_createDate, p.project_completionDate, p.project_lastActivityDate, t.team_id, t.team_name, t.team_createDate FROM project p JOIN team t on p.Team_team_id = t.team_id;”



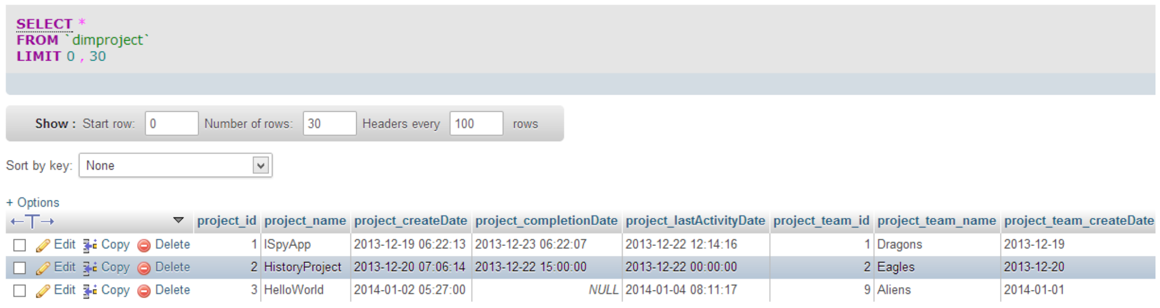
```
SELECT p.project_id, p.project_name, p.project_createDate, p.project_completionDate, p.project_lastActivityDate, t.team_id, t.team_name, t.team_createDate
FROM project p
JOIN team t ON p.Team_team_id = t.team_id
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Options

roject_id	project_name	project_createDate	project_completionDate	project_lastActivityDate	team_id	team_name	team_createDate
1	ISpyApp	2013-12-19 06:22:13	2013-12-23 06:22:07	2013-12-22 12:14:16	1	Dragons	2013-12-19
2	HistoryProject	2013-12-20 07:06:14	2013-12-22 15:00:00	2013-12-22 00:00:00	2	Eagles	2013-12-20
3	HelloWorld	2014-01-02 05:27:00	NULL	2014-01-04 08:11:17	9	Aliens	2014-01-01

Data warehouse: “SELECT * FROM dimProject”



```
SELECT *
FROM `dimproject`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

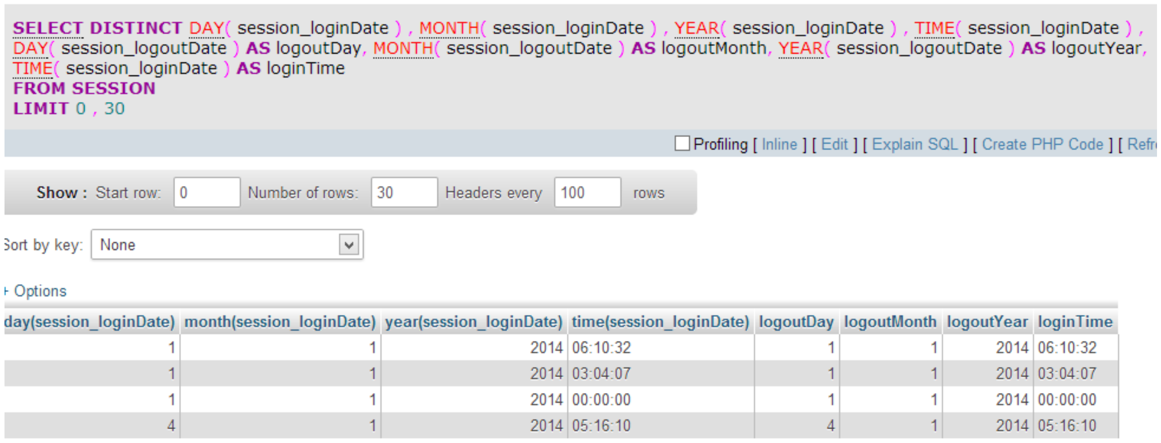
Sort by key: None

+ Options

←T→	project_id	project_name	project_createDate	project_completionDate	project_lastActivityDate	project_team_id	project_team_name	project_team_createDate
☐ Edit Copy Delete	1	ISpyApp	2013-12-19 06:22:13	2013-12-23 06:22:07	2013-12-22 12:14:16	1	Dragons	2013-12-19
☐ Edit Copy Delete	2	HistoryProject	2013-12-20 07:06:14	2013-12-22 15:00:00	2013-12-22 00:00:00	2	Eagles	2013-12-20
☐ Edit Copy Delete	3	HelloWorld	2014-01-02 05:27:00	NULL	2014-01-04 08:11:17	9	Aliens	2014-01-01

Physical Test #4:

Database: “SELECT DISTINCT DAY(session_loginDate), MONTH(session_loginDate) , YEAR(session_loginDate) , TIME(session_loginDate) , DAY(session_logoutDate) AS logoutDay, MONTH(session_logoutDate) AS logoutMonth, YEAR(session_logoutDate) AS logoutYear, TIME(session_loginDate) AS loginTime FROM SESSION;”



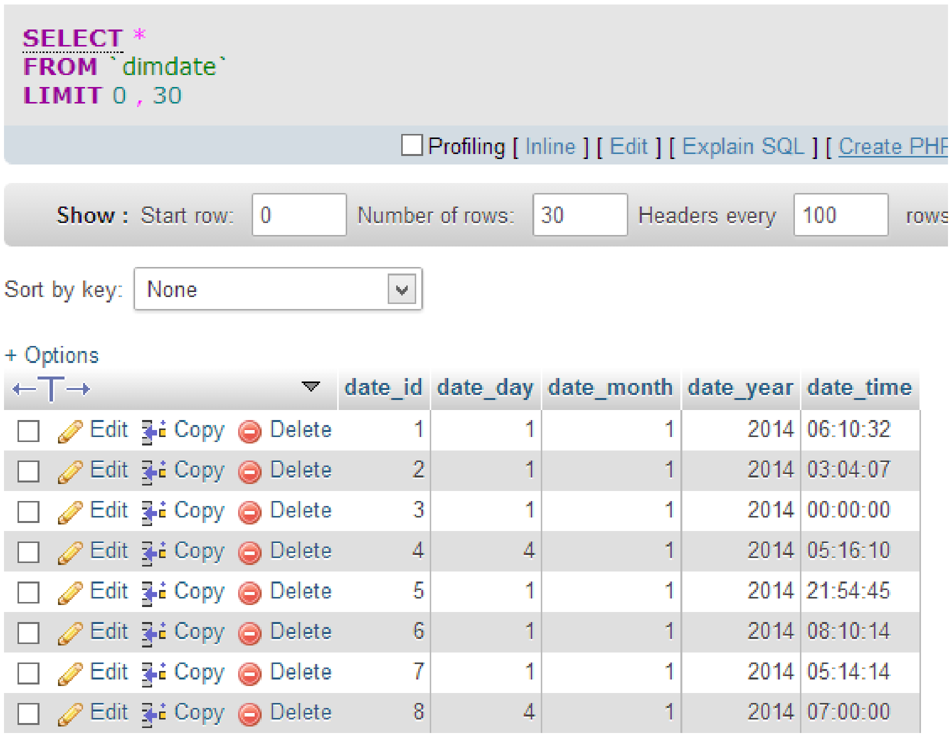
```
SELECT DISTINCT DAY( session_loginDate ) , MONTH( session_loginDate ) , YEAR( session_loginDate ) , TIME( session_loginDate ) ,
DAY( session_logoutDate ) AS logoutDay, MONTH( session_logoutDate ) AS logoutMonth, YEAR( session_logoutDate ) AS logoutYear,
TIME( session_loginDate ) AS loginTime
FROM SESSION
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

day(session_loginDate)	month(session_loginDate)	year(session_loginDate)	time(session_loginDate)	logoutDay	logoutMonth	logoutYear	loginTime
1	1	2014	06:10:32	1	1	2014	06:10:32
1	1	2014	03:04:07	1	1	2014	03:04:07
1	1	2014	00:00:00	1	1	2014	00:00:00
4	1	2014	05:16:10	4	1	2014	05:16:10

Data warehouse: “SELECT * FROM dimDate;”



```
SELECT *
FROM `dimdate`
LIMIT 0 , 30
```

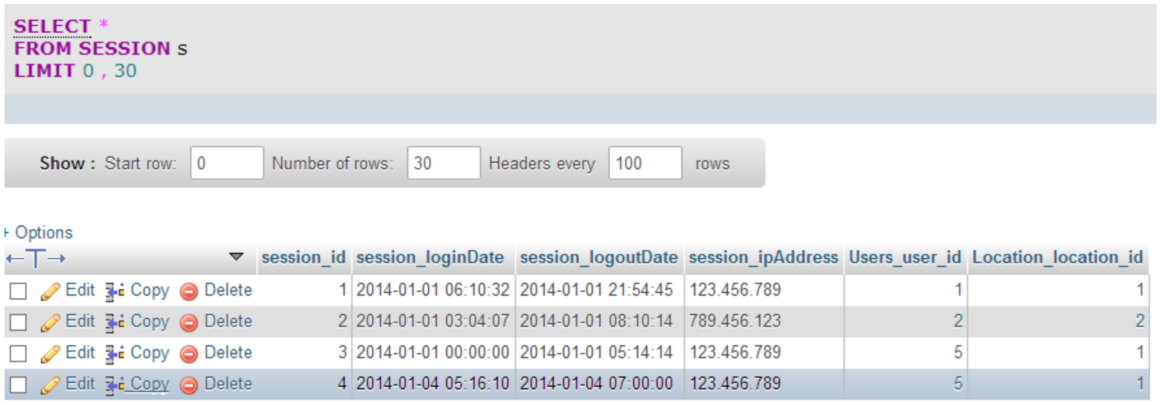
Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

date_id	date_day	date_month	date_year	date_time
1	1	1	2014	06:10:32
2	1	1	2014	03:04:07
3	1	1	2014	00:00:00
4	4	1	2014	05:16:10
5	1	1	2014	21:54:45
6	1	1	2014	08:10:14
7	1	1	2014	05:14:14
8	4	1	2014	07:00:00

Physical Test #5:

Database: “SELECT * FROM Session”



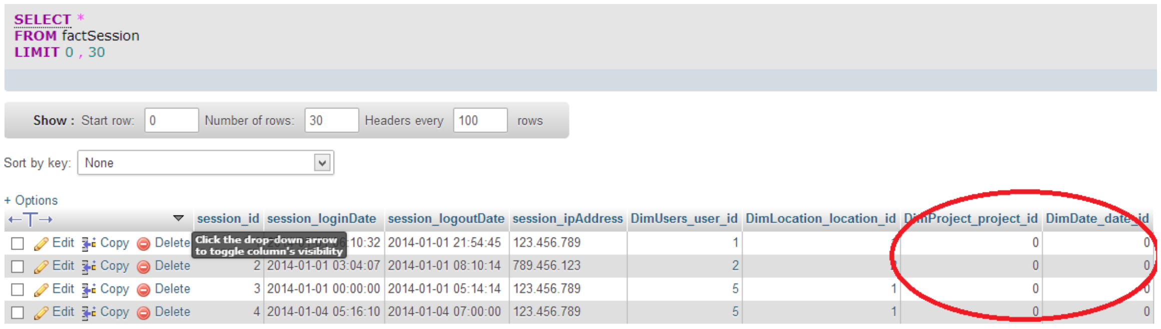
```
SELECT *
FROM SESSION s
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

	session_id	session_loginDate	session_logoutDate	session_ipAddress	Users_user_id	Location_location_id
Edit Copy Delete	1	2014-01-01 06:10:32	2014-01-01 21:54:45	123.456.789	1	1
Edit Copy Delete	2	2014-01-01 03:04:07	2014-01-01 08:10:14	789.456.123	2	2
Edit Copy Delete	3	2014-01-01 00:00:00	2014-01-01 05:14:14	123.456.789	5	1
Edit Copy Delete	4	2014-01-04 05:16:10	2014-01-04 07:00:00	123.456.789	5	1

Data warehouse: “SELECT * FROM factSession”



```
SELECT *
FROM factSession
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

Click the drop-down arrow to toggle column's visibility

	session_id	session_loginDate	session_logoutDate	session_ipAddress	DimUsers_user_id	DimLocation_location_id	DimProject_project_id	DimDate_date_id
Edit Copy Delete	[illegible]	[illegible]:10:32	2014-01-01 21:54:45	123.456.789	1	[illegible]	0	0
Edit Copy Delete	2	2014-01-01 03:04:07	2014-01-01 08:10:14	789.456.123	2	[illegible]	0	0
Edit Copy Delete	3	2014-01-01 00:00:00	2014-01-01 05:14:14	123.456.789	5	1	0	0
Edit Copy Delete	4	2014-01-04 05:16:10	2014-01-04 07:00:00	123.456.789	5	1	0	0

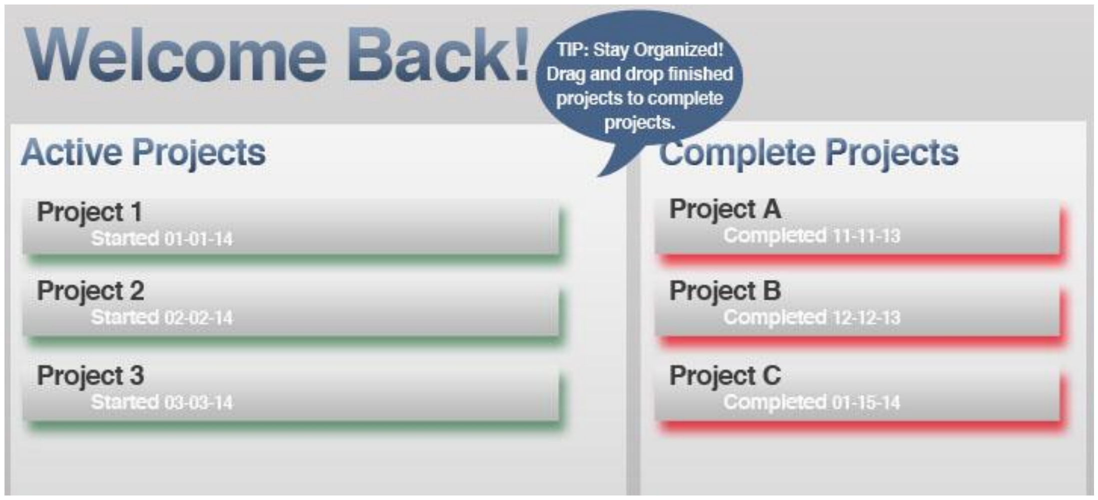
Note: DimProject_project_id & DimDate_date_id padded 0’s

CONSIDERATIONS

- Addition of “project_completionDate” allows for project throughput metric and provides a potential sales point for customer upgrade BUT requires user to indicate completion. This can be prompted for according to length of inactivity via “project_lastActivityDate” but is better handled with an Active/Inactive dashboard layout that allows for completion indication; a proper layout can provide the user a non-intrusive reminder on login to indicate project completion, creating

With a proper layout we can provide the user a non-intrusive reminder on login to indicate project completion that creates increased organization for the user, and an additional productivity metric for the organization.

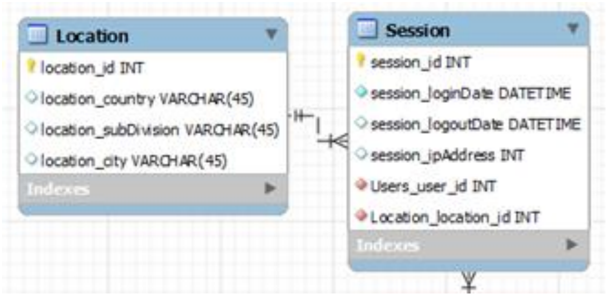
increased organization for the user, and an additional productivity metric for the organization.



- Database/data warehouse: "project_lastActivityDate" also allows for an email drip campaign to both prompt for indication of completion (providing a throughput metric) as well as a reminder to use the service

- Database/Data warehouse: Login/logout session length may be affected by timeout; if session length matches timeout setting, session data may be disregarded.
    - It is important to set a suitable timeout value that will rarely match a regular session length
    - Common timeout may indicate need for application redesign

- Application/Database: Location can/should be retrieved via ip address. This data should be available to the database as a product of the application itself; this process is assumed to not take place during ETL. Implementation may vary therefor for the purposes of this report we assume data is available to the original database


Location
location_id INT
location_country VARCHAR(45)
location_subDivision VARCHAR(45)
location_city VARCHAR(45)
Indexes
Session
session_id INT
session_loginDate DATETIME
session_logoutDate DATETIME
session_ipAddress INT
Users_user_id INT
Location_location_id INT
Indexes

- Database: Additional trigger/constraints may be required to enforce owner MUST remain on team that is owned by them; current trigger adds owner to team, but a change in "team_has_users" table allows for UPDATE

- ETL: AUTO INCREMENT may require to be reset by ETL if ETL is considered "new" each time and replaces old, for purposes of this project we assume this is not necessarily the case and we DROP and replace the data warehouse each time

- As no simple implementation exists to accurately LOAD factSession table with date_id and project_id data this has not been completed; foreign key constraints for these are not added, as they would be violated. Must add POST LOAD

Recommendation's

- Improvement of usage measure from simple login/logout; Application design should allow for some measure of engagement
    - Clicks per page
    - Activity tracking
    - Commenting
    - Task tracking
    - Track session activity by project; time spent on each, average number of projects accessed by session

See "Extended Model"

- Any additional features of the application, such as commenting or messaging among users or teams should be included; this provides further data regarding true engagement and these features may provide data indicative of the benefits of upgrading to the "moreThanAsesome" plan (selling point)

- Usage levels may be indicative of usage discrepancies such as location, gender or age; these discrepancies may indicate need for product redesign to increase accessibility to all ages or cater to differing demographics

- Additional information regarding user cancellation; "user_active" field and/or "user_cancellationDate" would provide valuable information on the number of cancellations, and these patterns could be drilled down to location, age, gender, or any other useful demographic

Extended Model

Current database design allows for scalability. Additional application features that may be easily added to database functionality that will generate new or more specific usage metrics are the result.

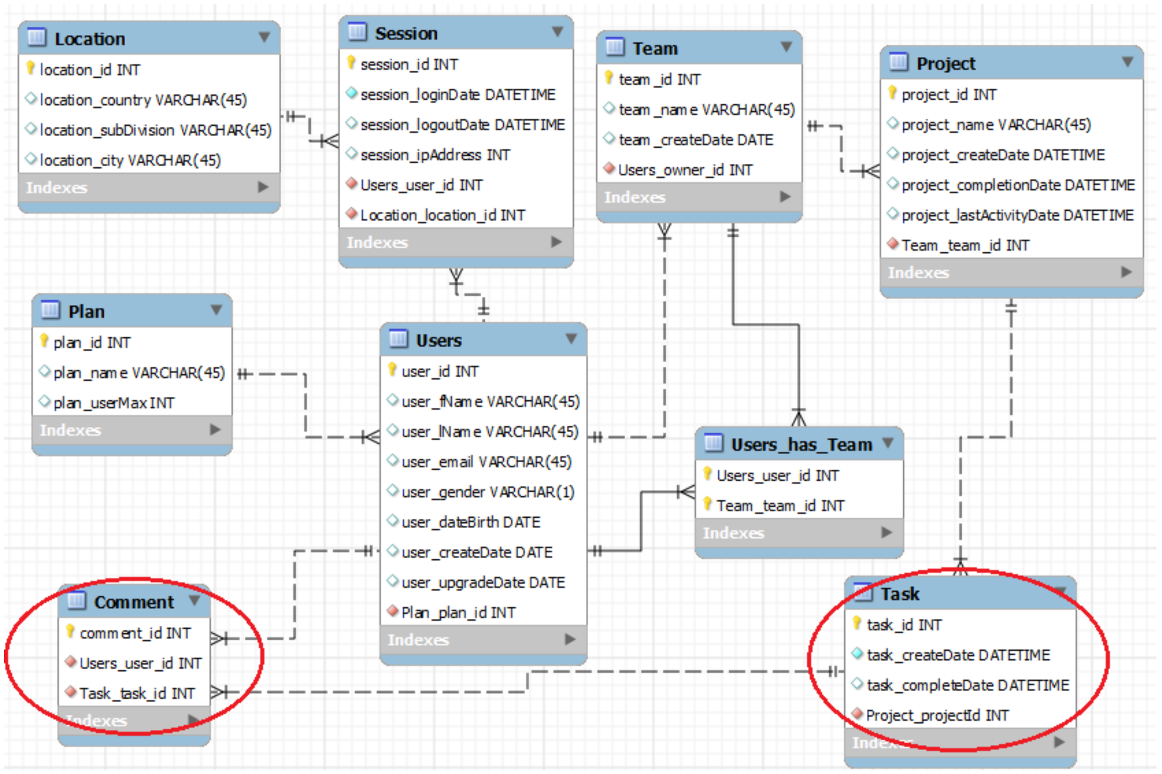
Additional Features

- Task tracking by project
- Task commenting

Additional Metrics

- Drill down to task throughput

- Track task activity by user
- Adds usage metric more indicative of engagement



Incomplete

- LOAD of “date_id” and “project_id” into “factSession” table of data warehouse
- ADD “date_id” and “project_id” FOREIGN CONSTRAINT; must be POST LOAD to maintain integrity of constraint, code to do so included in ETL script
- Creation of adequate amount of “dummy” data
- In depth unit testing of large amount of dummy data via QuerySurge (http://www.querysurge.com) or similar testing suite; physical testing with images include above