DATABASE FOR FACEBOOK

BUS243 – DATABASE MANAGEMENT

Submitted by:
Megha R Rao (SJSU ID: 013709488)
Rajasree Rajendran (SJSU ID: 013774358)
Sai Chaitanya Tolem (SJSU ID: 013008788)

TABLE OF CONTENTS

TABLE OF CONTENTS 2

DESCRIPTION OF THE DATA MODEL 4

QUERY DESCRIPTION & ANALYSIS 7

CONCLUSION 21

REFERENCES 24

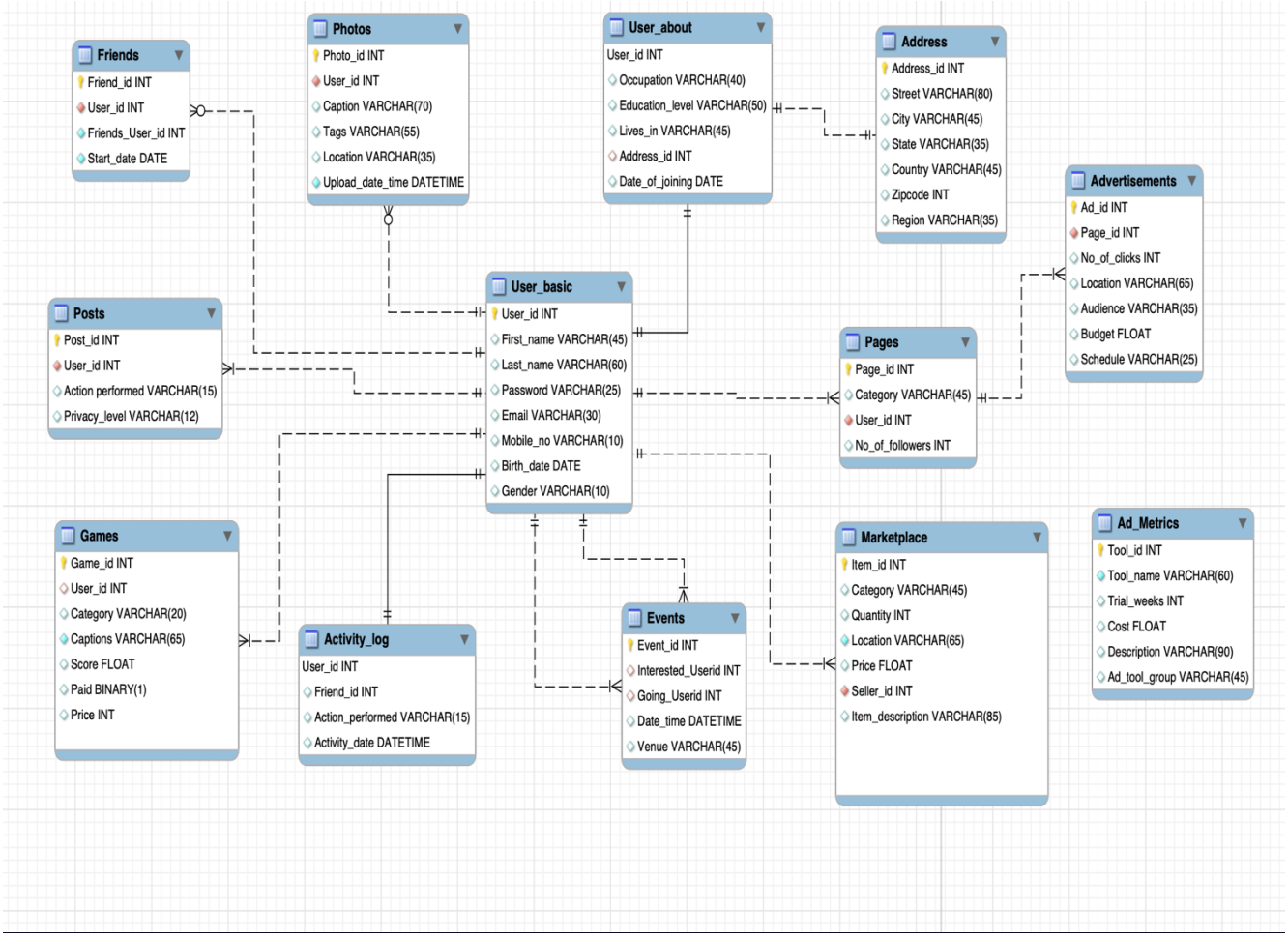
Overview

Facebook Inc. is an American social media and social networking company which was established in the year of 2004 by Mark Zuckerberg, Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. It was a school-based social network in Harvard University until 2006. In 2006, Facebook opened its doors to anyone older 13 years or older in the world. Facebook has a very user-friendly interface, and anyone with basic computer knowledge can use Facebook. The primary purpose of Facebook was to find friends who have not been in touch and to help them re-connect. Among the many other social networking sites, Facebook emerged to be the most successful one due to its unique features such as the 'Like' option, News Feed, Games and Business-friendly approach. The usage of Facebook has grown over time, the number of users crossing 2.27 billion monthly active users, as of September 2018, according to statistics. In the last decade, the use of all social networking sites has grown exponentially, with Facebook leading the list. This exponential growth means there is a huge amount of data available from all these users. Facebook was built around Big Data from its beginning, data was the driving force that kept it alive. But recently, Facebook has run into a lot of trouble for its usage of user data. Some attackers accessed personal data of at least 50 million Facebook users by exploiting a vulnerability in the system. This led to a huge uproar about the data usage of social networks, and increased privacy concerns among users, which led Facebook to shut down almost all of its open source data. In this project, we have attempted to create a database that is similar to that of Facebook, write queries to see how the database works and find some specific details using SQL queries such the most expensive game, a specific name search, etc.

Description of the Data Model

In order to proceed with this project, it is essential to understand how Facebook database works. Information on Facebook is mostly represented in the form of a social graph. The content is usually highly customizable based on the user's privacy settings. Hence, the data has to be stored in its original form and then filtered when needed. Facebook uses a combination of MySQL and Memcache for its database. Every user has his/her own dedicated database. Facebook uses MySQL because of its speed and reliability. Facebook stores friend relationships in a system called 'Tao' which uses MySQL. All tables have a hashed name and they are spread over a number of servers, similar to graph databases. Tao only stores the relations between entities. According to Facebook Inc., they collect information based on how a user uses their products. Information is collected from and about all computers and other devices the user uses to access Facebook, and this information is combined by them. The collected information is used to personalize features and content and to make suggestions for the user. The collected data is used to help advertisers to measure the effectiveness of their ads and services and to understand how the users interact with their services. They store the data until it is no longer necessary to provide the services or until the user deletes the account, whichever is first. While trying to understand the database structure of Facebook, we went through a number of resources and stumbled upon a resource where the class diagram was created by reverse engineering various Facebook business entities. Since only a very small portion of database details of Facebook is available as open source, we decided to create our database by reverse-engineering the the ER diagram we initially designed. Thereafter, we generated data ourselves before proceeding to integrate the data in sql. We went way beyond the presribed 5 problems/queries when we decided to tackle around 20 problems/queries. Some were routine tasks whereas others were aimed at analysing by querying the database.

"Without a systematic way to start and keep data clean, bad data will happen"- Donato Diorio



ER Diagram for Facebook Database

Creating the Entity Relationship (ER) diagram was the most fun part while doing this project. We initially thought this would be an easier job, but the unique issue about dealing with Facebook data was that, it was all over the internet, but nowhere specific for open source use. After referring to many resources, we realized that News Feed does not have a relation to the other entities. The above ER diagram represents the Facebook profile database of a single user as an entity.

The ER diagram has the following entities with their own attributes:

⇒ ***User_basic***: Has various basic attributes of the user namely *User_id* (Primary Key), *First_name, Last_name, Password, Email, Mobile number, Birth_date* and *Gender.*

⇒ ***User_about:*** with attributes *Occupation, Education, Lives_in, Address_id* and *Date_of_joining.*

⇒ ***Address:*** Attributes are *Address_id, Street, City, State, Country, Zip code* and *Region*.

⇒ ***Pages:*** Attributes are *Page_id, Category, User_id, No_of_followers.*

⇒ ***MarketPlace:*** Attributes are *item_id, category, quantity, location, price, seller_id* and *item_description*.

⇒ ***Events:*** Attributes are *event_id, interested_userid, Going_userid, Date_time, Venue.*

⇒ ***Photos:*** Attributes are *photo_id, user_id, caption, tags, location, upload_date_time*.

⇒ ***Activity_log:*** with attributes *User_id, friend_id, action_performed, activity_date*.

⇒ ***Games:*** attributes are *game_id, user_id, category, captions, score, paid.*

⇒ ***Posts:*** with attributes *post_id, user_id, action_performed, privacy_level.*

⇒ ***Friends:*** with attributes *friend_id, user_id, friends_user_id, category, start_date, since_when.*

⇒ ***Advertisements:*** *ad_id, page_id, no_of_clicks, traffic, location, audience, budget, schedule.*

⇒ ***DA Toolkit:*** *tool_id, tool_name, trial_weeks, cost, description.*

Query description & Analysis

Based on the above data, we decided to find open source Facebook user data in order to do the queries in mySQL. Unfortunately, it was very hard to find open datasets for Facebook, due to all the recent data breach incidents Facebook Inc. has gone through. After a great amount of research, we decided to create a Facebook database on our own with some fictitious data. Thus, data preparation was done. Since the fictitious data was in Google Sheets format, we converted it into csv format and then later, converted the data to sql format and uploaded to mySQL workbench. The following are the queries we performed based on our prepared data:

⇒ Our first goal was to create tables and entities to accommodate the data. After obtaining the sql data, we used CREATE function to create tables. One such was to create a table called 'Sales_table' which shows all the items ordered by price and quantity from the 'Marketplace' table. The following query was used to create the table:

CREATE TABLE Sales_Table AS (SELECT Item_id, Category, Price, Quantity FROM marketplace);

```
CREATE TABLE Sales_Table AS
(SELECT Item_id, Category, Price, Quantity FROM marketplace);
```

This statement creates a table called 'Sales_Table' with entities from 'Marketplace'.

⇒ After creating the Sales Table, we went ahead and calculated which items were being sold the most, with the following statement:

SELECT * FROM Sales_Table
WHERE Quantity IN (SELECT MAX(Quantity) FROM Sales_Table)
ORDER BY Price DESC;

Output:

```
SELECT * FROM Sales_Table
WHERE Quantity IN (SELECT MAX(Quantity) FROM Sales_Table)
ORDER BY Price DESC;
```

Filter Rows: | Export: | Wrap Cell Content:

Item_id	Category	Price	Quantity
500501	Cell Phones & Accessories	000000000168	0000000030
500508	Home & Garden	000000000067	0000000030

⇒ We also calculated the total number of items currently in marketplace along with the numbers in each category.

SELECT category, quantity, count(category) FROM marketplace GROUP BY category WITH rollup;

Output:

```
SELECT category "Category", COUNT(category) "Item count", quantity "Stock quantity"
FROM marketplace GROUP BY category WITH ROLLUP
ORDER BY quantity;
```

Filter Rows: | Export: | Wrap Cell Content:

Category	Item count	Stock quantity
Camera & Photo	1	0000000011
Major Appliances	1	0000000012
Office Products	1	0000000014
Music	1	0000000016

⇒ Since we wanted to know the total cost distributed for each category, we did the following query and sorted the results in 5 quantile ranges of price.

SELECT item_id, category, quantity*price AS cost, NTILE (5) OVER (ORDER BY quantity*price) AS quantile FROM marketplace GROUP BY category;

Output:

```
SELECT item_id,category AS "Item category", quantity*price AS "Cost",
NTILE(5) OVER (ORDER BY quantity*price) AS "Quantile"
FROM marketplace GROUP BY category;
```

Filter Rows: | Export: | Wrap Cell Content:

item_id	Item category	Cost	Quantile
500514	Musical Instruments	1150	1
500512	Major Appliances	1740	1
500509	Independent Design	1962	1
500508	Home & Garden	2010	1
500500	Camera & Photo	2079	2

⇒ As a part of performing routine activities, we calculated the percentage of female users and male users in our dataset.

SELECT ROUND(((SELECT COUNT(*) FROM user_basic WHERE gender = 'Male') / (SELECT COUNT(*) FROM user_basic))*100,2) AS "Percentage of Male users",ROUND(((SELECT COUNT(*) FROM user_basic WHERE gender = 'Female') /(SELECT COUNT(*) FROM user_basic))*100,2) AS "Percentage of Female users" FROM DUAL;

Output:

```
SELECT ROUND(((SELECT COUNT(*) FROM user_basic WHERE gender = 'Male')/
(SELECT COUNT(*) FROM user_basic))*100,2) AS "Percentage of Male users",
ROUND(((SELECT COUNT(*) FROM user_basic WHERE gender = 'Female')/
(SELECT COUNT(*) FROM user_basic))*100,2) AS "Percentage of Female users"
FROM DUAL;
```

Filter Rows: | Export: | Wrap Cell Content:

Percentage of Male users	Percentage of Female users
45.16	54.84

⇒ Next was the calculation of Age of users as a derived attribute.

SELECT *, YEAR(CURDATE()) - YEAR(birth_date) AS Age, NTILE(4) OVER (ORDER BY 'Age' ASC) AS "Quantile (Age groups - 1 to 4)" FROM user_basic;

Output:

```
SELECT *, YEAR(CURDATE()) - YEAR(birth_date) AS Age,
NTILE(4) OVER (ORDER BY 'Age' ASC) AS "Quantile (Age groups - 1 to 4)"
FROM user_basic;
```

Filter Rows: | Export: | Wrap Cell Content:

User_id	First_name	Last_name	Password	Email	Mobile_no	Birth_date	Gender	Age	Quantile (Age groups - 1 to 4)
100504	Bryan	Custock	Bryan456	bryan456@abc.com	123-100-111	1990-01-10	Male	28	1
100505	Candy	Antony	Candy456	candy456@xyz.com	123-100-100	1991-01-15	Female	27	1
100506	Christian	Bernard	Christian678	christian678@xyz.com	100-111-123	1992-01-20	Male	26	1
100507	Curt	Morgan	Curt678	curt678@abc.com	111-100-123	1992-01-30	Male	26	1
100508	David	Sheperd	Dave123	dave123@xyz.com	100-123-111	1991-02-01	Male	27	2
100509	Dolphia	Nandi	Dolphia234	dolphia@abc.com	111-123-123	1991-02-10	Female	27	2
100510	Dwight	Eisenhower	Dwight456	dwight456@abc.com	100-110-100	1993-02-15	Male	25	2
100511	Eleanor	Phoebe	Eleanor123	eleanor123@xyz.com	110-110-110	1990-03-01	Female	28	2
100512	Ethan	Bartizal	Ethan234	ethan234@abc.com	110-100-110	1991-03-05	Male	27	2

⇒ In order to execute a realistic scenario, we decided to find the users who are from California, and then found their education details.

SELECT concat (First_name," ", Last_name) as Name, education_level, lives_in, occupation FROM user_basic JOIN user_about ON user_basic.user_id = user_about.user_id WHERE lives_in = 'California';

Output:

```
SELECT CONCAT(first_name," ",last_name) AS Name, Education_level, Occupation, Lives_in
FROM user_basic JOIN user_about ON user_basic.user_id = user_about.user_id
WHERE lives_in ='California';
```

Filter Rows: | Export: | Wrap Cell Content:

Name	Education_level	Occupation	Lives_in
Aaron Abraham	Undergraduate	Accountant	California
Abhishek Acharya	High School	Broker	California
Harry Potter	Professional Degree	Lawyer	California
Hermione Granger	Undergraduate	Consultant	California

⇒ As a part of the routine activities, we decided to find out a specific user's friends. We chose the user "Harry Potter" and tried to find out the friends of Mr. Potter. Even though we were expecting Ron Weasley and Hermione Granger to be in the list, we were quite surprised by the results.

SELECT user_basic.user_id, friends.start_date, CONCAT (user_basic.first_name," ",user_basic.last_name) AS Friend, (CURDATE() - friends.start_date) AS FriendshipinDays FROM friends JOIN user_basic ON user_basic.user_id = friends.friends_user_id WHERE friends.user_id = (SELECT user_id FROM user_basic WHERE user_basic.first_name = 'Harry');

Output:

```
1  SELECT user_basic.user_id, friends.start_date, CONCAT (user_basic.first_name," ",user_basic.last_name) AS Friend,
2  (CURDATE() - friends.start_date) AS FriendshipinDays FROM friends JOIN user_basic ON user_basic.user_id = friends.friends_user_id
3  WHERE friends.user_id = (SELECT user_id FROM user_basic WHERE user_basic.first_name = 'Harry');
```

100% 96:3

Result Grid | Filter Rows: Search | Export:

user_id	start_date	Friend	FriendshipinDays
100509	2018-10-03	Dolphia Nandi	207
100510	2018-10-18	Dwight Eisenhower	192
100511	2018-09-28	Eleanor Phoebe	282

⇒ In order to find the details of an event, we created a query with respect to event entity. The following query finds the count of all events listed:

- **SELECT Venue AS "Event Venue", Date_Time, COUNT(Invitees_Userid) AS "No. of Invites" FROM Events GROUP BY Venue, Date_Time;**

Output:

```
SELECT Venue AS "Event Venue", Date_Time, COUNT(Invitees_Userid) AS "No. of Invites"
FROM Events GROUP BY Venue, Date_Time;
```

Filter Rows: | Export: | Wrap Cell Content:

Event Venue	Date_Time	No. of Invites
The Palazzo, Martin Street	2018-11-07 11:11:00	7
ICC Durban Arena	2018-10-28 11:10:00	6
National Center For The Performing Arts	2018-10-18 11:10:00	5
Ticketpro Dome	2018-09-28 11:09:00	5
Ticketpro Dome	2018-09-18 11:09:00	1
Loftus Versfeld Stadium	2018-09-18 11:09:00	1
Nissan Sports Centre	2018-09-18 11:09:00	1
Cadillac Arena	2018-09-18 11:09:00	1
Nelson Mandela Bay Stadium	2018-09-18 11:09:00	2

⇒ We found the names of the invitees to the events:

SELECT CONCAT(first_name," ",last_name) AS Name, Venue, Date_Time FROM user_basic JOIN events ON events.invitees_userid = user_basic.user_id ORDER BY Venue, Date_Time;

Output:

```
SELECT CONCAT(first_name," ",last_name) AS Name, Venue, Date_Time
FROM user_basic JOIN events ON events.invitees_userid = user_basic.user_id
ORDER BY Venue, Date_Time;
```

Filter Rows: | Export: | Wrap Cell Content:

Name	Venue	Date_Time
Eleanor Phoebe	Cadillac Arena	2018-09-18 11:09:00
Pansy Parkinson	Loftus Versfeld Stadium	2018-09-18 11:09:00
Harry Potter	Nelson Mandela Bay Stadium	2018-09-18 11:09:00
Brinda Lakshman	Nelson Mandela Bay Stadium	2018-09-18 11:09:00
Stan Shunpike	Nissan Sports Centre	2018-09-18 11:09:00
James Potter	Ticketpro Dome	2018-09-18 11:09:00
Jaya Devi	Ticketpro Dome	2018-09-28 11:09:00

⇒ Among its millions of users, there are a lot of inactive users who could be considered as shadow profiles/ inactive users. We executed a query to find out who were the users who had not logged in the past six months, thus classifying them as inactive users. We deleted such users without compromising the referential integrity.

SET SQL_SAFE_UPDATES = 0;
DELETE FROM user_basic WHERE user_id IN
(SELECT u.user_id FROM User_about u JOIN activity_log a ON a.User_id = u.User_id
WHERE (a.Activity_date < (NOW() - INTERVAL 6 MONTH)) GROUP BY u.user_id);
SET SQL_SAFE_UPDATES = 1;
DELETE FROM user_basic WHERE user_id = 100528;

Output:

```
SET SQL_SAFE_UPDATES = 0;
DELETE FROM user_basic WHERE user_id IN
(SELECT u.user_id FROM User_about u JOIN activity_log a ON a.User_id = u.User_id
WHERE (a.Activity_date < (NOW() - INTERVAL 6 MONTH)) GROUP BY u.user_id);
SET SQL_SAFE_UPDATES = 1;
```

```
9  08:24:26  DELETE FROM user_basic WHERE user_id IN (SELECT u.user_id FROM User_...  6 row(s) affected
```

```
DELETE FROM user_basic WHERE user_id = 100528;
```

```
13  08:36:07  DELETE FROM user_basic WHERE user_id = 100528  1 row(s) affected
```

⇒ We imagined a scenario where the FBI asks Facebook team to help them out by finding the users whose names end with "**er**", which could help them in an ongoing investigation. We decided to help them out by doing this query:

SELECT * FROM user_basic WHERE Last_name LIKE '%er';

Output:

```
1  SELECT * FROM user_basic WHERE Last_name LIKE '%er';
2
```

100% 53:1

Result Grid | Filter Rows: Search | Edit: | Export/Import:

User_id	First_name	Last_name	Password	Email	Mobile_no	Birth_date	Gen...
100510	Dwight	Eisenhower	Dwight456	dwight456@abc.com	100-110-100	1993-02-15	Male
100516	Harry	Potter	Harry456	harry456@abc.com	200-200-210	1992-07-11	Male
100517	Hermione	Granger	Hermione...	hermione123@abc...	200-210-210	1992-03-15	Fem...
100520	James	Potter	James456	james456@abc.com	211-210-211	1975-04-20	Male
100522	Lilly	Potter	Lilly456	lilly456@abc.com	211-210-200	1980-06-10	Fem...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

⇒ The FBI seemed to have obtained an anonymous tip on the person, the tip was that the name ends with “otter”, which luckily narrows down the suspects.

SELECT *from user_basic WHERE Last_name LIKE '%otter%';

Output:

```
SELECT * FROM user_basic WHERE last_name LIKE '%otter';
```

Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

User_id	First_name	Last_name	Password	Email	Mobile_no	Birth_date	Gender
100516	Harry	Potter	Harry456	harry456@abc.com	200-200-210	1992-07-11	Male
100520	James	Potter	James456	james456@abc.com	211-210-211	1975-04-20	Male
100522	Lilly	Potter	Lilly456	lilly456@abc.com	211-210-200	1980-06-10	Female

⇒ We decided to do a descriptive analysis of budget allocation of Advertisements by finding total, minimum, maximum and average of budgets.

SELECT COUNT(budget) AS 'Total No.', SUM(budget) AS 'Total Budget', AVG(budget) AS 'Average Budget', MIN(budget) AS 'Minimum', MAX(budget) AS 'Maximum Bdget' FROM advertisements;

Output:

```
SELECT COUNT(budget) AS 'Total No.', SUM(budget) AS 'Total Budget',
AVG(budget) AS 'Average Budget', MIN(budget) AS 'Minimum Budget',
MAX(budget) AS 'Maximum Budget' FROM advertisements;
```

Filter Rows: | Export: | Wrap Cell Content:

Total No.	Total Budget	Average Budget	Minimum Budget	Maximum Budget
19	300475	15814.473684210527	11388	19967

⇒ After finding out the details about the budget, we explored various mathematical functins such as midrange, standard deviation, harmonic average, arithmetic mean and geometric mean.

SELECT ((MAX(budget) + MIN(budget))/2) AS 'Midrange', STDDEV(budget) AS 'Standard Deviation', VARIANCE(budget) AS 'Variance', SUM(budget) / COUNT(budget) AS 'Arithmetic Mean', COUNT(budget) / SUM(1/budget) AS 'Harmonic Average', EXP(SUM(LOG(budget))) AS 'Geometric Mean' FROM advertisements;

Output:

```
SELECT ((MAX(budget) + MIN(budget))/2) AS 'Midrange', STDDEV(budget) AS 'Standard Deviation',
VARIANCE(budget) AS 'Variance', SUM(budget) / COUNT(budget) AS 'Arithmetic Mean',
COUNT(budget) / SUM(1/budget) AS 'Harmonic Average', EXP(SUM(LOG(budget))) AS 'Geometric Mean'
FROM advertisements;
```

Filter Rows: | Export: | Wrap Cell Content:

Midrange	Standard Deviation	Variance	Arithmetic Mean	Harmonic Average	Geometric Mean
15677.5	2496.081399828...	6230422.354570637	15814.473684210...	15411.398394690914	4.7497378573801775e79

⇒ Next, as a part of our objective, we did recommendations of least expensive games from the Games data.

SELECT name AS ' Are you ready? - Exciting games!', category 'Category', paid as 'Do I pay?', price AS 'Recommended - Low to High Price!' FROM games ORDER BY price ASC;

Output:

```
1  SELECT name AS 'Are you ready? - Exciting Games!', category 'Category', paid as 'Do I pay?',
2  price AS 'Recommended - Low to High Price!'
3  FROM games ORDER BY price ASC;
4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Are you ready? - Exciting Games!	Category	Do I pay?	Recommended - Low to High Price!
8 Ball Pool	Simulation	No	0
Farm Heroes Saga	Real-life	No	0
Criminal Case	Puzzle	No	0
8 Ball Pool	Simulation	No	0
Candy Crush	Puzzle	No	0
8 Ball Pool	Simulation	No	0
Soccer Manager 2018	Simulation	No	0
Candy Crush	Puzzle	No	0
Epic Winning quest	Adventure	Yes	2
Wordscapes	Crossword	Yes	2
Farm Heroes Saga	Real-life	Yes	3
Angry birds	Adventure	Yes	3.99
Winning Slots	Slots game	Yes	3.99
Criminal Case	Puzzle	Yes	4.99
Epic Winning quest	Adventure	Yes	5.99
Dragon City	Action	Yes	7

⇒ As per our objectives of the project, we planned to create views, and we created views for Ad plans.

CREATE VIEW Ads_Premium ASSELECT Tool_name AS "Ad Tools you get!", Description AS "Details", Cost AS "Price - only from",Trial_weeks AS "Trial weeks" FROM ad_metrics ORDER BY Cost;

CREATE VIEW Ads_Booster ASSELECT Tool_name AS "Ad Tools you get!", Description AS "Details", Cost AS "Price - only from",Trial_weeks AS "Trial weeks" FROM ad_metrics WHERE Ad_tool_group = 1 OR Ad_tool_group = 2 ORDER BY Cost;

CREATE VIEW Ads_Basic ASSELECT Tool_name AS "Ad Tools you get!", Description AS "Details", Cost AS "Price - only from",Trial_weeks AS "Trial weeks" FROM ad_metrics WHERE Ad_tool_group = 1 ORDER BY Cost;

Output:

```
CREATE VIEW Ads_Premium AS
SELECT Tool_name AS "Ad Tools you get!", Description AS "Details",  Cost AS "Price - only from",
Trial_weeks AS "Trial weeks" FROM ad_metrics ORDER BY Cost;

CREATE VIEW Ads_Booster AS
SELECT Tool_name AS "Ad Tools you get!", Description AS "Details",  Cost AS "Price - only from",
Trial_weeks AS "Trial weeks" FROM ad_metrics WHERE Ad_tool_group = 1 OR Ad_tool_group = 2 ORDER BY Cost;

CREATE VIEW Ads_Basic AS
SELECT Tool_name AS "Ad Tools you get!", Description AS "Details",  Cost AS "Price - only from",
Trial_weeks AS "Trial weeks" FROM ad_metrics WHERE Ad_tool_group = 1 ORDER BY Cost;
```

⇒ One of our goals was to predict user interests. For this we used the following query by joining three tables, thus doing the prediction of user interests.

```
SELECT a.user_id AS "User", p.post_id AS "Recommended post",
a.friend_id AS "Related to friend", a.action_performed AS "Friend's action"
FROM activity_log a INNER JOIN friends f ON a.user_id = f.friends_user_id
INNER JOIN posts p ON f.friends_user_id = p.user_id
WHERE p.privacy_level <> 'Only Me';
```

Output:

```
SELECT a.user_id AS "User", p.post_id AS "Recommended post",
a.friend_id "Related to friend", a.action_performed AS "Friend's action"
FROM activity_log a INNER JOIN friends f ON a.user_id = f.friends_user_id
INNER JOIN posts p ON f.friends_user_id = p.user_id
WHERE p.privacy_level <> 'Only Me';
```

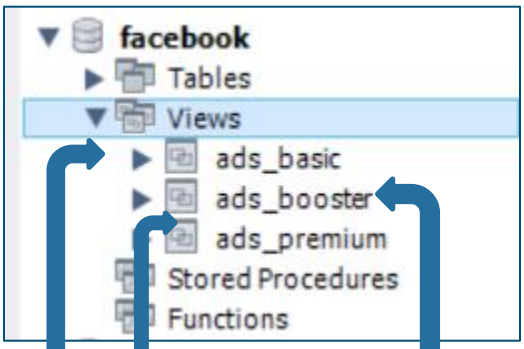
Filter Rows: | Export: | Wrap Cell Content:

User	Recommended post	Related to friend	Friend's action
100524	400622	300524	Commented
100524	400623	300524	Commented
100525	400624	300525	Shared
100524	400622	300524	Commented
100524	400623	300524	Commented
100525	400624	300525	Shared
100500	400600	300500	Liked
100504	400608	300504	Saved
100504	400612	300504	Saved
100504	400608	300504	Saved
100504	400612	300504	Saved
100503	400601	300503	Shared

⇒ After creating the views, we could view the Views with Data Analysis toolkits for each Ad plan.

SELECT * FROM Ads_Basic;
SELECT * FROM Ads_Booster;
SELECT * FROM Ads_Premium;

Output:




```
1 •  SELECT * FROM Ads_Basic;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ad Tools you get!	Details	Price - only from	Trial weeks
Impressions	This is defined as the number of times y...	15	2
Placement	The Placement chart shows your ad's p...	20	12
Clicks	This is defined as the number of times y...	25	3

```
1 •  SELEC * FROM Ads_Booster
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Ad Tools you get!	Details	Price - only from	Trial weeks
Impressions	This is defined as the number of times yo...	15	2
Placement	The Placement chart shows your ad's per...	20	12
Clicks	This is defined as the number of times yo...	25	3
Conversions Metric	Conversions refer to actions taken on you...	30	3
Performance	The Performance chart shows the numbe...	32	12
Demographics	The Demographics chart shows how your...	35	12

```
1 •  SELECT * FROM Ads_Premium;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

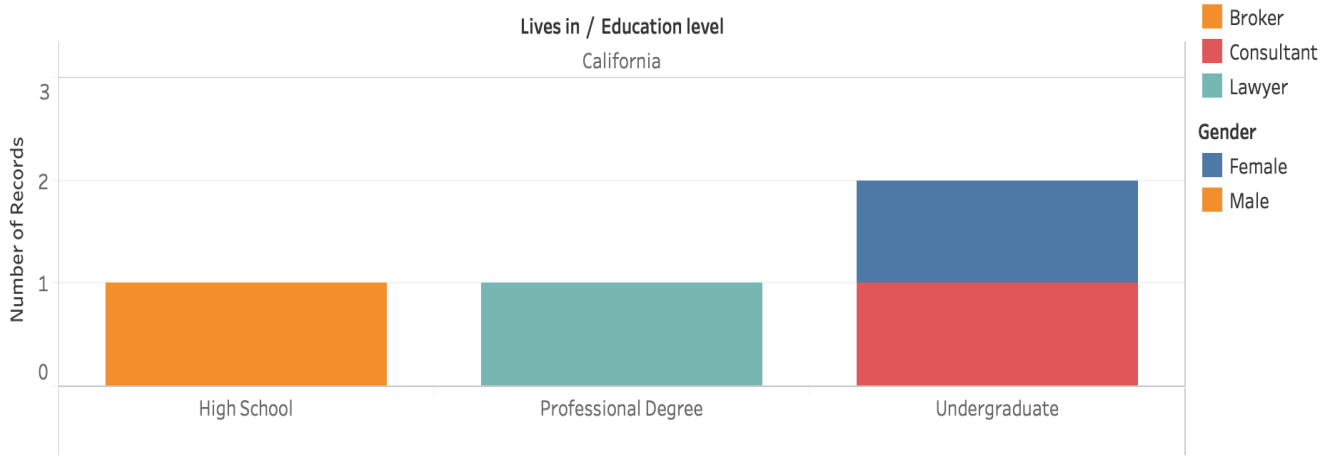
Ad Tools you get!	Details	Price - only from	Trial weeks
Impressions	This is defined as the number of times yo...	15	2
Placement	The Placement chart shows your ad's per...	20	12
Clicks	This is defined as the number of times yo...	25	3
Conversions Metrics	Conversions refer to actions taken on you...	30	3
Performance	The Performance chart shows the numbe...	32	12
Demographics	The Demographics chart shows how your...	35	12
Lifetime Value	Lifetime value is the projected revenue t...	35	4
Return on Ad Spend (ROAS)	Return on ad spend (ROAS) is defined as ...	45	2
Split testing	Use split tests—also known as A/B tests—...	50	2
Attribution	Assign conversion credit to marketing tou...	55	4

CONCLUSION

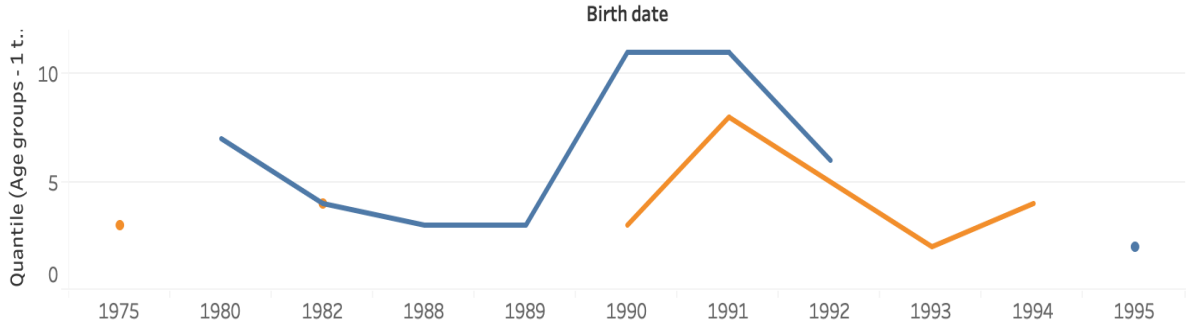
As a part of the project, we were able to manually create a database, run queries on the database and find results using various queries we learnt in class. We were able to fulfill most of our project objectives and learned a lot in the process. The objectives were to create tables and entities, and perform routine Facebook activities. The database we built satisfies all the required entities for performing a variety of queries based on user preference. In an organization like Facebook, where database and extraction of data from database plays a major role, it is imperative to work with tools like MySQL and help in carrying out required operations. Using the queries like the ones used in this project, it helps to narrow down data from 2.2 billion monthly users. Queries which use functions such as CREATE, JOIN, SELECT, UPDATE, DELETE are all most common queries used in a scenario like that of Facebook's. During this project, we faced a number of challenges : one of them being the unavailability of open datasets for Facebook users. We rectified the same by recreating databases on our own. The next challenge we faced was converting the csv files to sql format, many online tools were tried in vain. Thanks to Professor Shirani for guiding us during the challenging times, as per Professor Shirani's instructions, we were able to successfully convert the csv files to sql files using SQLite Studio software. We believe we were able to fulfill most of our objectives and we did learn a lot during the process.

TABLEAU DASHBOARD :

Background study of California residents



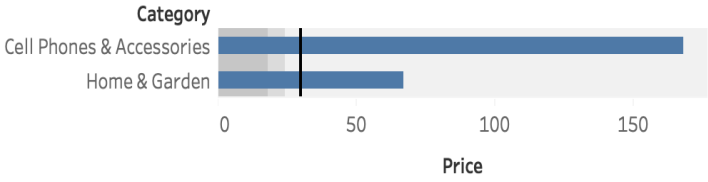
Age group analysis



Percentage of Female & Male users

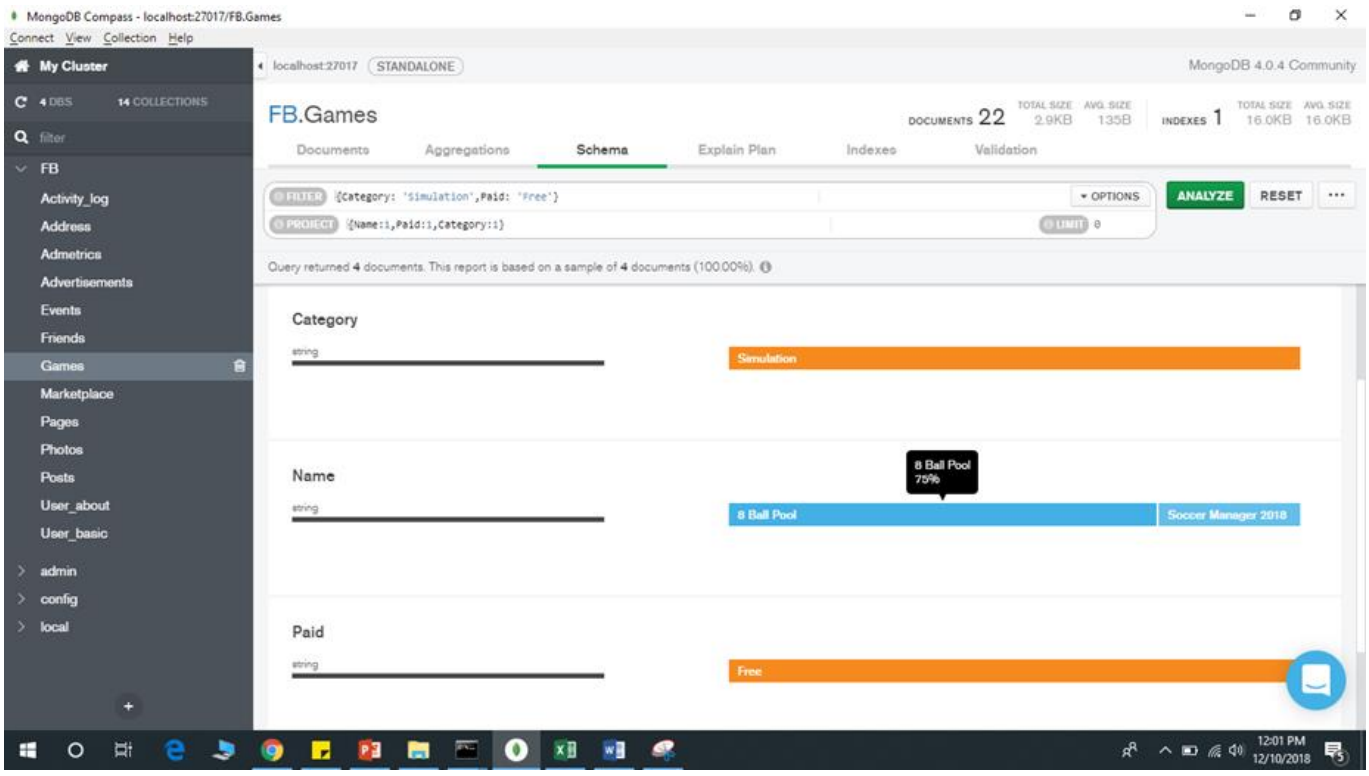
Percentage of Female use..	54.840
Percentage of Male users	45.160

Maximum sold items in Marketplace



MongoDB Analysis:

Of the games which are free under the simulation category - Most people prefer to play 8 Ball Pool (75%) – using the Analyze Schemas feature in MongoDB.



REFERENCES

- http://web.archive.org/web/20121031052327/http://blogs.x2line.com/al/archive/2007/06/02/3124.aspx
- https://www.facebook.com/full_data_use_policy
- https://www.usenix.org/conference/atc13/technical-sessions/presentation/bronson
- https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
- https://www.facebook.com/notes/facebook-engineering/tao-the-power-of-the-graph/10151525983993920/
- https://www.makeuseof.com/tag/facebook-work-nuts-bolts-technology-explained/