Final Project

Skouzos Paschalis

7.1 Presentation

The application attempts to mimic the database in the game distribution software Steam by Valve. The information about the games was parsed for a previous midterm project with a program I wrote from steamdb.info. The database is required in order to build and manage the store. I have added a referrer system where users can refer new people to the store and as a reward they get discounts according to how many people they have invited. The second discount system in effect depends on the age of each game, the longer a game is on the platform the higher its discount. I have also supposed that the company had created hype before the opening of the store and all the accounts were created before that in order not to have account create date problems inconsistencies (I have omitted that field). Finally, there are roles that impose restrictions on what different accounts can do(CEO, store manager, expansion manager, users).

7.2 Table Description

Games

One of the two main tables, it stores 100 games and provides

game_id, game_name, game_developer, game_score(rating), game_price, game_publish_date

The link for the steam store webpage can be produced by

http://store.steampowered.com/app/ID

Genres

The genres for each game. It had to be a different table because one game may have more than one genres. It is important to store them because the users may want to search for games in a specific genre and also the company can monitor the tendencies of the market.

game_id, genre

Countries

This table contains the countries in which the store is available and is used in order to make sure a user outside from those countries can't register but also to relate country codes with country names. Country codes in the users table were used in order to save space. In addition, we have the GMT time offset for each country.

country_code, country_name

Users

The second main table which contains the users of our application. It is imperative to bookkeep that information not only for communication with our customers but for data analysis. It contains

user_id, user_username, user_first_name, user_last_name, user_email, country_code, user_date_of_birth, user_referrer_id, user_total_actv_mins

Friends

This is the table that stores which users are friends with whom. The two fields are

friend1_id, friend2_id

Purchases

The table that contains the purchases by the users. Each row contains one game bought by one user and the date of purchase. The primary key is the purchase id and the reason the primary key isn't the user id combined with the game id is because a user can buy it more times in order to gift it to a friend. Because of the varying discounts the price isn't included here. The fields are

purchase_id, user_id, game_id, purchase_date

Referrer Discounts

The table that holds the amount of discount a user gets according to how many people have claimed that he is their referrer. The discount_start field indicates the minimum invited people he should have in order to get the corresponding discount. The fields are

discount_start, discount_percent

Activities

This table contains discrete activities for each purchase. The fields are

Purchase_id, actv_duration_mins, actv_datetime

Aged Game Discounts

The table that holds the amount of discount a game gets depending on its age. The fields are

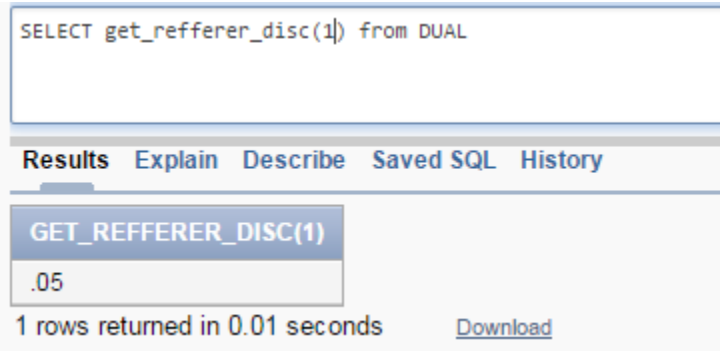
aged_game_discount_age, aged_game_discount_percent

7.3 Enhancements

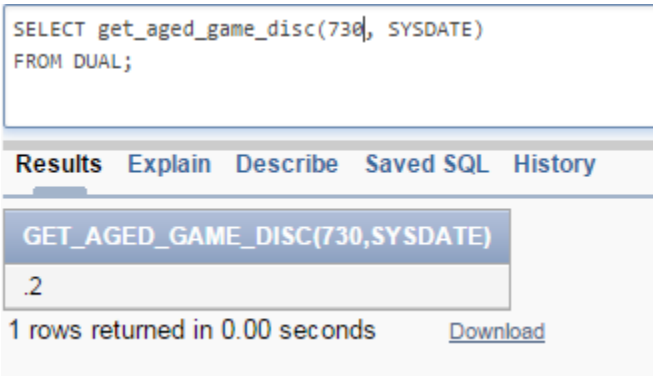
There are 3 major enhancements to the midterm project. The first one is the addition of the table with the discounts for each game based on its age. The second is the alternation of the activities table which used to be cumulative for each user but now it holds separate activities (when and for how much a user interacted with a purchase). The last one is the addition of the user_total_actv_mins column to the users table which holds the total amount of time a user has spent on his purchases. A minor enhancement is renaming the user_country_code to country_code in order to have consistency with the primary key it is referencing.

7.4 Queries

Q1. Create a function that returns the % discount of a user based on the amount of referrers he has.



Q2. Create a function that returns the % discount for a game based on the age of that game.



Q3. All our times are stored in GMT+0 time. Create a function that returns the local time of a user.

```
SELECT get_user_time(1, TO_DATE('2016-02-03 04:27:00','YYYY-MM-DD HH24:MI:SS'))
FROM DUAL;
```

Results Explain Describe Saved SQL History

GET_USER_TIME(1,TO_DATE('2016-02-0304:27:00','YYYY-MM-DDHH24:MI:SS'))
05:27:00

User with ID 1 is from Germany so GMT+1 we expect 05:27:00 and the result is correct.

Q4. Add a trigger to activities in order to update the total time a user has spent playing games.

The results after uploading 7.insertActivities.sql

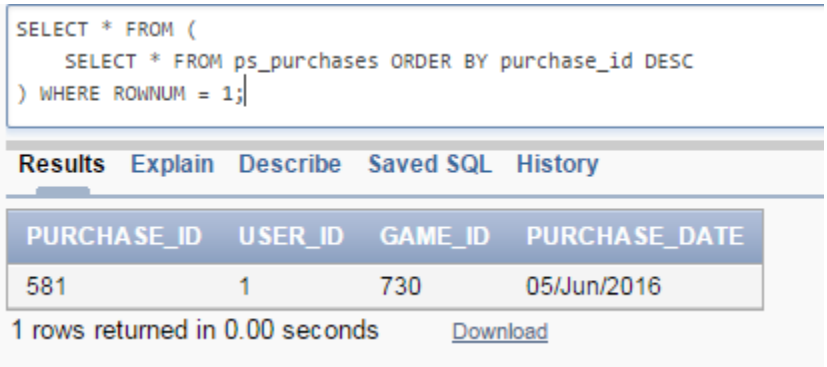
```
SELECT * FROM ps_users;
```

Results Explain Describe Saved SQL History

USER_ID	USER_USERNAME	USER_FIRST_NAME	USER_LAST_NAME	USER_EMAIL	COUNTRY_CODE	USER_DATE_OF_BIRTH	USER_REFERRER_ID	USER_TOTAL_ACTV_MINS
1	espears	Eva	Spears	eva.spears@mail.com	DE	25/Oct/1981	-	1067
2	carterb	Carter	Black	black@gmail.com	CR	25/Jul/1991	1	347
3	ecalhoun	Ellie	Calhoun	calhoun@mail.com	ES	29/Jun/1981	2	865
4	bgoff	Benjamin	Goff	goff@gmail.com	IT	20/Aug/1986	2	109
5	gholloway	Grayson	Holloway	graysonholloway@mail.com	GR	23/Mar/1983	3	576
6	tcampos	Thomas	Campos	thomascampos@mail.com	GR	09/Aug/1989	5	478
7	cooperr	Cooper	Reese	reese@mail.com	GR	20/Apr/1986	1	1042
8	levio	Levi	Osborne	leviosborne@yahoo.com	NO	14/Mar/1990	1	541
9	lydiaw	Lydia	Wall	lydia.wall@mail.com	BR	04/Nov/1988	1	317
10	oreilly	Owen	Reilly	owen.reilly@mail.com	IT	25/Aug/1969	1	930
11	morganb	Morgan	Bowers	morganbowers@gmail.com	US	27/Aug/1987	-	134
12	bentleyv	Bentley	Vance	vance@gmail.com	FR	02/Apr/1978	11	640
13	mcarroll	Makayla	Carroll	carroll@mail.com	RU	14/Jul/1967	-	180
14	lreilly	Lucy	Reilly	lucyreilly@yahoo.com	TR	27/Nov/1976	6	451
15	camilab	Camila	Boyer	boyer@gmail.com	CR	20/Jan/1983	8	0
16	kevinc	Kevin	Christensen	kevinchristensen@mail.com	IT	08/Oct/1973	-	619

Q5. The table ps_purchases has an auto-increment style primary key(purchase_id) but when we created the database we forgot to implement it and now we have the last PK being 580. Create a trigger so we don't have to manually enter a PK when we are inserting into ps_purchases.

Testing screenshot.



```
SELECT * FROM (
    SELECT * FROM ps_purchases ORDER BY purchase_id DESC
) WHERE ROWNUM = 1;
```

Results Explain Describe Saved SQL History

PURCHASE_ID	USER_ID	GAME_ID	PURCHASE_DATE
581	1	730	05/Jun/2016

1 rows returned in 0.00 seconds Download

Q6. Create a function that returns the price of a purchase then create a view of purchases with the price.

```
SELECT * FROM purchases_with_price;
```

Results Explain Describe Saved SQL History

PURCHASE_ID	USER_ID	GAME_ID	PURCHASE_DATE	PRICE
307	55	8500	24/Nov/2015	11.994
308	55	23600	12/Aug/2014	5.994
309	55	130	23/Oct/2015	2.994
310	56	130	29/Dec/2014	2.994
311	56	2820	01/Jul/2015	9.594
312	56	8340	13/Dec/2014	11.994
313	57	22000	06/Feb/2014	5.994
314	57	20500	19/Oct/2015	11.994
315	57	4530	10/Feb/2014	5.994
316	57	18070	20/Dec/2015	2.998
317	57	10	29/Oct/2014	5.994
318	57	2810	05/Aug/2014	5.994
319	57	7860	27/Jul/2014	2.994
320	57	21690	03/Dec/2015	11.994
321	57	23600	05/Dec/2015	5.994
322	57	8500	03/Feb/2014	11.994

Q7. Show the percentage distribution of the time of date that users start playing games(based on their local time) in order for the company to know at what time of day to improve the server capacity.

```
WITH total_actvs AS (SELECT COUNT(*) total_actvs FROM ps_activities)
SELECT SUBSTR(get_user_time(p.user_id, a.actv_datetime),0,2)||':00' "FROM",
          (TO_NUMBER(SUBSTR(get_user_time(p.user_id, a.actv_datetime),0,2))+1)||':00' "TO",
          COUNT(*)*100/(SELECT total_actvs FROM total_actvs) || '%' "PERCENTAGE"
FROM ps_activities a
NATURAL JOIN ps_purchases p
GROUP BY SUBSTR(get_user_time(p.user_id, a.actv_datetime),0,2)
ORDER BY SUBSTR(get_user_time(p.user_id, a.actv_datetime),0,2);
```

Results Explain Describe Saved SQL History

FROM	TO	PERCENTAGE
00:00	1:00	1.3%
01:00	2:00	1.4%
02:00	3:00	1.2%
03:00	4:00	1.7%
04:00	5:00	1.4%
05:00	6:00	1.8%
06:00	7:00	2.4%
07:00	8:00	3.8%
08:00	9:00	5.2%
09:00	10:00	6.5%
10:00	11:00	8.1%
11:00	12:00	10.4%
12:00	13:00	7.6%
13:00	14:00	8.3%
14:00	15:00	9.7%
15:00	16:00	7.1%
16:00	17:00	6.5%
17:00	18:00	4.3%
18:00	19:00	3.4%
19:00	20:00	3%
20:00	21:00	1.3%
21:00	22:00	1.2%
22:00	23:00	1.3%
23:00	24:00	1.1%

24 rows returned in 0.09 seconds Download

From the results we see that there is a normal distribution of the time of day the users are playing, so the company should have higher server capacity from 10:00 to 15:00.

6.4 Further development

Everything I could think of as an enhancement in my midterm project report has been implemented except the account creation date.