

Sample Database Problems and Solutions

Using the DVD_Rental sample database from www.postgresqltutorial.com, I have devised a practice problem and solution for the DVD rental company.

Some inconsistencies that were found during my analysis include the following:

-The rental date (rental.rental_date) and payment date (payment.payment_date) are inconsistent. The payment occurs around nine months after the rental takes place. The rental and payment are also sometimes completed by an employee at another store.

To best view this inconsistency, I suggest creating a table by joining attributes, or columns, from the payment and rental tables into a new diff_summary table.

Rental table attributes: rental_id, rental_date as date_rented and rented_by.

Payment table attributes: payment_id, processed_by ,and payment_date.

This information will tell us which employee rented out the DVD and which employee processed the payment. Ideally, the payments would be processed by the same employee who rented the DVD out.

Diff_Summary Code

```
1. CREATE TABLE diff_summary (  
2.     payment_id          INT,  
3.     payment_date        TIMESTAMP,  
4.     processed_by        INT,  
5.     rented_by           INT,  
6.     date_rented         TIMESTAMP,  
7.     rental_id           INT  
8. );  
9.
```

A brief sample of the data supports my claim of inconsistency:

	payment_id [PK] integer	payment_date timestamp without time zone	processed_by integer	rented_by integer	date_rented timestamp without time zone	rental_id integer
1	22344	2007-03-21 05:03:07.996577	1	2	2005-08-21 06:34:41	14277
2	30052	2007-04-08 16:36:39.996577	1	2	2005-07-08 18:08:13	4835
3	26264	2007-04-11 07:09:13.996577	2	1	2005-07-11 08:40:47	6140
4	24416	2007-03-20 18:01:47.996577	1	1	2005-08-20 19:33:21	13994
5	26745	2007-04-27 03:57:23.996577	2	2	2005-07-27 05:28:57	7109
6	18803	2007-02-21 01:40:47.996577	2	2	2005-06-21 03:12:21	3248
7	31297	2007-04-07 23:21:01.996577	2	2	2005-07-08 00:52:35	4479
8	29007	2007-04-27 12:59:08.996577	2	1	2005-07-27 14:30:42	7346
9	31789	2007-04-30 05:50:18.996577	1	1	2005-07-30 07:21:52	9061
10	29089	2007-04-11 00:36:55.996577	2	2	2005-07-11 02:08:29	6019
11	20549	2007-03-01 18:07:34.996577	1	1	2005-08-01 19:39:08	10738
12	26770	2007-04-09 03:31:01.996577	2	2	2005-07-09 05:02:35	5073
13	23000	2007-03-01 03:27:56.996577	2	1	2005-08-01 04:59:30	10335

We can expect payment IDs and rental IDs to differ, but the payment date and the date the DVD was rented should be the same or at least within a few days of one another. The data above shows that the DVDs were rented in 2005 and the payment was taken in 2007-a two-year gap. You'll also notice that some of the payments were processed by the same employee, and other times the payments were processed by another employee. This fictional company only has two store locations and two employees. Each employee is assigned one location. Therefore, it isn't possible for one employee to complete the rental while the other employee processes the payment two years later.

Using the below query returns the diff_summary table data listed by payment_date in ascending order.

```
1. SELECT * FROM diff_summary AS d
2. ORDER BY d.payment_date, d.processed_by, d.payment_id, d.rented_by, d.date_rented, d.rental_id;
3.
```

The SELECT query returns similar information; the employees cannot possibly be in two places simultaneously; therefore, the table remains inconsistent.

	payment_id [PK] integer	payment_date timestamp without time zone	processed_by integer	rented_by integer	date_rented timestamp without time zone	rental_id integer
1	17793	2007-02-14 21:21:59.996577	2	2	2005-06-14 22:53:33	1158
2	18173	2007-02-14 21:23:39.996577	2	1	2005-06-14 22:55:13	1159
3	19399	2007-02-14 21:29:00.996577	1	2	2005-06-14 23:00:34	1160
4	18441	2007-02-14 21:41:12.996577	2	1	2005-06-14 23:12:46	1163
5	18698	2007-02-14 21:44:52.996577	1	1	2005-06-14 23:16:26	1164
6	19498	2007-02-14 21:44:53.996577	2	2	2005-06-14 23:16:27	1165
7	18686	2007-02-14 21:45:29.996577	1	1	2005-06-14 23:17:03	1166
8	18051	2007-02-14 22:03:35.996577	2	1	2005-06-14 23:35:09	1168
9	19036	2007-02-14 22:11:22.996577	2	2	2005-06-14 23:42:56	1169
10	18456	2007-02-14 22:16:01.996577	2	2	2005-06-14 23:47:35	1170
11	19239	2007-02-14 22:23:12.996577	2	2	2005-06-14 23:54:46	1173
12	18870	2007-02-14 22:41:17.996577	2	2	2005-06-15 00:12:51	1174
13	19265	2007-02-14 22:43:41.996577	2	2	2005-06-15 00:15:15	1175

The stakeholders of the DVD Rental company want to update their library. They'd like to know which films are rented the least and which films are the most popular. I created two new tables to properly display this data: a least_popular table and a most_popular table. But tables use and show the same data in different ways. The least_popular table shows the top 10 least popular movies, while the most_popular table shows the top 10 most popular movies.

Most Popular Films

```
1. CREATE TABLE most_popular (
2. film_id SMALLINT PRIMARY KEY,
3. film_title VARCHAR(255),
4. genre VARCHAR(25),
5. times_rented INT);
6.
```

```
1. INSERT INTO most_popular (film_id, film_title, genre, times_rented)
2.
3. SELECT f.film_id, f.title AS film_title, c.name AS genre, COUNT(r.rental_id) AS times_rented
4.
5. FROM film AS f
6.
```

```

7. INNER JOIN inventory AS i
8.     ON f.film_id=i.film_id
9.
10. INNER JOIN film_category AS fc
11.     ON f.film_id=fc.film_id
12.
13. INNER JOIN category AS c
14.     ON c.category_id=fc.category_id
15.
16. INNER JOIN rental AS r
17.     ON i.inventory_id=r.inventory_id
18.
19. GROUP BY c.name, f.film_id;
20.

```


Most Popular Films Top 10

```

1. select * FROM most_popular
2. WHERE times_rented > 30
3. ORDER BY times_rented DESC
4. LIMIT 10;

```

Returns

	film_id [PK] smallint 	film_title character varying (255) 	genre character varying (25) 	times_rented integer 
1	103	Bucket Brotherhood	Travel	34
2	738	Rocketeer Mother	Foreign	33
3	489	Juggler Hardly	Animation	32
4	331	Forward Temple	Games	32
5	382	Grit Clockwork	Games	32
6	730	Ridgemont Submarine	New	32
7	767	Scalawag Duck	Music	32
8	735	Robbers Joon	Children	31
9	621	Network Peak	Family	31
10	418	Hobbit Alien	Drama	31

Bucket Brotherhood appears to be the current most rented movie.

Least Popular Films

```

1. CREATE TABLE least_popular (
2. film_id SMALLINT PRIMARY KEY,
3. film_title VARCHAR(255),
4. genre VARCHAR(25),
5. times_rented INT);
6.

```

Least Popular Films Top 10

```

1. INSERT INTO least_popular (film_id, film_title, genre, times_rented)
2.
3. SELECT f.film_id, f.title AS film_title, c.name AS genre, COUNT(r.rental_id) AS times_rented
4.
5. FROM film AS f
6.

```

```

7. INNER JOIN inventory AS i
8.     ON f.film_id=i.film_id
9.
10. INNER JOIN film_category AS fc
11.     ON f.film_id=fc.film_id
12.
13. INNER JOIN category AS c
14.     ON c.category_id=fc.category_id
15.
16. INNER JOIN rental AS r
17.     ON i.inventory_id=r.inventory_id
18.
19. GROUP BY c.name, f.film_id;
20.

```

Returns

	film_id [PK] smallint	film_title character varying (255)	genre character varying (25)	times_rented integer
1	428	Homicide Peach	Family	22
2	299	Factory Dragon	Travel	17
3	12	Alaska Phantom	Music	26
4	113	California Birds	Sports	12
5	945	Virginian Pluto	Documentary	29
6	699	Private Drop	Games	5
7	28	Anthem Luke	Comedy	15
8	445	Hyde Doctor	Classics	26
9	710	Rage Games	Family	19
10	209	Darkness War	Drama	13
11	521	Lies Treatment	Drama	27

Most Popular Genre of Top 10 Most Popular Films

```

1. select * FROM most_popular
2. WHERE times_rented > 30
3. ORDER BY genre DESC
4. LIMIT 10;
5.

```

Returns

	film_id [PK] smallint	film_title character varying (255)	genre character varying (25)	times_rented integer
1	103	Bucket Brotherhood	Travel	34
2	369	Goodfellas Salute	Sci-Fi	31
3	730	Ridgemont Submarine	New	32
4	767	Scalawag Duck	Music	32
5	331	Forward Temple	Games	32
6	382	Grit Clockwork	Games	32
7	738	Rocketeer Mother	Foreign	33
8	621	Network Peak	Family	31
9	31	Apache Divine	Family	31
10	753	Rush Goodfellas	Family	31

While the top rented movie was of the Travel genre, there are more rented movies in the Family genre.

Least Popular Genre of Top 10 Least Popular Films

```
1. SELECT * FROM least_popular
2. WHERE times_rented < 6
3. ORDER BY genre DESC
4. LIMIT 10;
5.
```

Returns

There were seventeen total film titles that had been rented less than six times. The top ten are below. Depending on the cost of storing DVD rental inventory, the stakeholders may want to remove the film titles generated from the above query.

	film_id [PK] smallint	film_title character varying (255)	genre character varying (25)	times_rented integer
1	903	Traffic Hobbit	Travel	5
2	612	Mussolini Spoilers	Sports	5
3	558	Mannequin Worst	New	5
4	904	Train Bunch	Horror	4
5	699	Private Drop	Games	5
6	310	Fever Empire	Games	5
7	362	Glory Tracy	Games	5
8	781	Seven Swarm	Games	5
9	584	Mixed Doors	Foreign	4
10	459	Informer Double	Foreign	5

Most Popular Genre Rented

```
1. CREATE TABLE popular_genre (
2. genre VARCHAR(25) PRIMARY KEY,
```

```

3. times_rented INT);
4.
5. INSERT INTO popular_genre (genre, times_rented)
6. SELECT c.name AS genre, COUNT(r.rental_id) AS times_rented
7. FROM category AS c
8. INNER JOIN film_category AS fc ON c.category_id=fc.category_id
9. INNER JOIN film AS f ON f.film_id=fc.film_id
10. INNER JOIN inventory AS i ON f.film_id=i.film_id
11. INNER JOIN rental AS r ON i.inventory_id=r.inventory_id
12. GROUP BY genre
13. ;
14.
15. SELECT * FROM popular_genre
16. ORDER BY times_rented DESC;

```

Returns

	genre [PK] character varying (25)	times_rented integer
1	Sports	1179
2	Animation	1166
3	Action	1112
4	Sci-Fi	1101
5	Family	1096
6	Drama	1060
7	Documentary	1050
8	Foreign	1033
9	Games	969
10	Children	945
11	Comedy	941
12	New	940
13	Classics	939
14	Horror	846
15	Travel	837
16	Music	830

While the Travel genre had the most rented movie, the Travel genre ranks number 15 out of the 16 movie genres available from the DVD Rental service. If the DVD Rental company wants to invest in the most popular genre, the data suggests that the DVD Rental company should invest in the Sports genre. According to the Least Popular Movie Top Ten results, one would think that the Game and Foreign genres were rented the least. This is proven false according to the above data. The Travel and Music genres were rented the least number of times.

The stakeholders are worried about inactivity and would like to send coupons to inactive members to persuade them to rent DVDs from the DVD Rental company again. This marketing tactic would require viewing the customer and payment data from our DVD Rental database. To mail coupons, the stakeholders request the customer's email, address, city, district, postal code, and country.

Inactivity Coupon Table

Returns data with all the customers and their activity status.

```

1. CREATE TABLE inactivity_coupon (

```

```

2. customer_id int PRIMARY KEY,
3. active INT,
4. total_spent numeric(5,2),
5. full_name VARCHAR(70),
6. email VARCHAR(50),
7. address VARCHAR(50),
8. city VARCHAR(50),
9. state VARCHAR(20),
10. postal_code VARCHAR(10),
11. country VARCHAR(50)
12. );
13.
14. INSERT INTO inactivity_coupon (
15. customer_id, active, total_spent, full_name, email, address, city, state, postal_code, country)
16. SELECT cu.customer_id, cu.active, SUM(p.amount) AS total_spent, CONCAT(cu.first_name, ' ', cu.last_name) AS
full_name, cu.email, a.address, ci.city, a.district AS state, a.postal_code, co.country
17. FROM customer AS cu
18. INNER JOIN payment AS p ON cu.customer_id=p.customer_id
19. INNER JOIN address AS a ON cu.address_id=a.address_id
20. INNER JOIN city AS ci ON a.city_id=ci.city_id
21. INNER JOIN country AS co ON ci.country_id=co.country_id
22. GROUP BY cu.customer_id, cu.active, a.address, ci.city, a.district, a.postal_code, co.country;
23.

```

Only Inactive Customers (Ordered by most spent to least spent)

```

1. SELECT * FROM inactivity_coupon
2. WHERE active=0
3. ORDER BY total_spent DESC;
4.

```

	customer_id [PK] integer	active integer	total_spent numeric (5,2)	full_name character varying (70)	email character varying (50)	address character varying (50)	city character varying (50)	state character varying (20)	postal_code character varying (10)	country character varying (50)
1	368	0	139.69	Harry Arce	harry.arce@sakilacustomer.org	1922 Miraj Way	Najafabad	Esfahan	13203	Iran
2	558	0	135.72	Jimmie Eggleston	jimmie.eggleston@sakilacustomer.org	505 Madiun Boulevard	Wroclaw	Dolnoslaskie	97271	Poland
3	482	0	125.74	Maurice Crawley	maurice.crawley@sakilacustomer.org	1785 So Bernardo do Campo Street	Coatzacoalcos	Veracruz	71182	Mexico
4	406	0	121.69	Nathan Runyon	nathan.runyon@sakilacustomer.org	264 Bhimavaram Manor	Charlotte Amalie	St Thomas	54749	Virgin Islands, U.S.
5	592	0	111.71	Terrance Roush	terrance.roush@sakilacustomer.org	42 Fontana Avenue	Szkesfehrv	Fejr	14684	Hungary
6	241	0	110.68	Heidi Larson	heidi.larson@sakilacustomer.org	1103 Bilbays Parkway	Xiangfan	Hubei	87660	China
7	16	0	109.75	Sandra Martin	sandra.martin@sakilacustomer.org	360 Toulouse Parkway	Southend-on-Sea	England	54308	United Kingdom
8	446	0	109.72	Theodore Culp	theodore.culp@sakilacustomer.org	1704 Tambaram Manor	Uluberia	West Bengali	2834	India
9	510	0	102.77	Ben Easter	ben.easter@sakilacustomer.org	886 Tonghae Place	Kamyin	Volgograd	19450	Russian Federation
10	64	0	91.70	Judith Cox	judith.cox@sakilacustomer.org	1966 Amroha Avenue	Daxian	Sichuan	70385	China
11	169	0	86.80	Erica Matthews	erica.matthews@sakilacustomer.org	1294 Firozabad Drive	Pingxiang	Jiangxi	70618	China
12	534	0	81.78	Christian Jung	christian.jung@sakilacustomer.org	949 Allende Lane	Amroha	Uttar Pradesh	67521	India
13	315	0	67.86	Kenneth Gooden	kenneth.gooden@sakilacustomer.org	1542 Lubumbashi Boulevard	Bat Yam	Tel Aviv	62472	Israel
14	124	0	57.86	Sheila Wells	sheila.wells@sakilacustomer.org	848 Tafuna Manor	Ktahya	Ktahya	45142	Turkey
15	271	0	56.84	Penny Neal	penny.neal@sakilacustomer.org	1675 Xiangfan Manor	Kumbakonam	Tamil Nadu	11763	India

From the returned data, the stakeholders would learn that only fifteen inactive customers exist. The table shows the stakeholders where to mail and the DVD rental coupons as well as how much each customer has spent with the DVD Rental company. Depending on their marketing strategy, the stakeholders may choose to give higher-valued coupons to the customers who have spent the least to entice them into returning and spending more.