Migration 013: Add Credits to Customer Stats View

Overview

This migration updates the `customer_stats` view to include credit balance and transaction information for each customer. This allows the admin dashboard to display credit information without additional database queries.

What This Migration Does

1. Drops the existing `customer_stats` view
2. Recreates the view with additional fields:
   - `credit_balance`: Current credit balance for the customer
   - `total_credits_purchased`: Total credits purchased by the customer
   - `total_credits_used`: Total credits used by the customer
   - Also includes previously missing fields: `contact_email`, `max_users`, `logo_url`

How to Apply This Migration

Option 1: Using the Node.js Migration Script

```
cd /home/ubuntu/roblox-tool
node scripts/run-migration.js 013
```

Option 2: Using psql (Direct Database Access)

```
psql -U your_username -d verifylens_db -f database/migrations/
013_add_credits_to_customer_stats.sql
```

Option 3: Manual Application via psql

```
# Connect to your database
psql -U your_username -d verifylens_db

# Then run:
\i database/migrations/013_add_credits_to_customer_stats.sql
```

Verification

After applying the migration, you can verify it worked by running:

```sql
-- Check if the view has the new columns
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'customer_stats'
AND column_name IN ('credit_balance', 'total_credits_purchased', 'total_credits_used')
;

-- Test the view
SELECT id, name, credit_balance, total_credits_purchased, total_credits_used
FROM customer_stats
LIMIT 5;
```

Rollback

If you need to rollback this migration, you can restore the old view:

```sql
DROP VIEW IF EXISTS customer_stats;

CREATE OR REPLACE VIEW customer_stats AS
SELECT
    c.id,
    c.name,
    c.is_active,
    c.created_at,
    COUNT(DISTINCT u.id) as total_users,
    COUNT(DISTINCT CASE WHEN u.is_active THEN u.id END) as active_users,
    COUNT(sh.id) as total_searches,
    MAX(sh.searched_at) as last_search_at,
    MAX(u.last_login) as last_login_at
FROM customers c
LEFT JOIN users u ON c.id = u.customer_id
LEFT JOIN search_history sh ON c.id = sh.customer_id
GROUP BY c.id, c.name, c.is_active, c.created_at;
```

Impact

- **Zero downtime**: This is a view update, not a table modification
- **No data loss**: This only changes how data is queried, not stored
- **Backward compatible**: The view still includes all previous columns
- **Performance**: Minimal impact, as the view uses LEFT JOIN with indexed tables

Notes

- This migration is safe to run on production
- The view is recreated immediately, no need to restart the application
- All existing queries using `customer_stats` will continue to work
- New queries can now access credit information directly from the view