Admin Credit Addition Fix Guide

Problem Summary

When trying to add credits through the admin dashboard, the system was throwing an error:

```
Failed to add credits
Details: invalid input syntax for type integer: "MANUAL_ADMIN_2_1764551502843"
```

Root Cause

The `credit_transactions` table has a `payment_id` column that was defined as `INTEGER`, but the admin credit addition code generates string-based payment IDs in the format `MANUAL_ADMIN_{userId}_{timestamp}`.

This mismatch caused PostgreSQL to reject the insertion because it couldn't convert the string to an integer.

Solution

We've created **Migration 014** to fix this issue by:

1. **Removing the foreign key constraint** that linked `payment_id` to `stripe_payments.id`
2. **Changing the column type** from `INTEGER` to `TEXT`
3. **Adding a validation constraint** to ensure payment IDs follow valid formats
4. **Adding an index** for better query performance

How to Apply the Fix

Option 1: Supabase SQL Editor (Recommended)

1. Open your Supabase project dashboard
2. Navigate to **SQL Editor** in the left sidebar
3. Click **New query**
4. Copy and paste the contents of `database/migrations/SUPABASE_MIGRATION_014.sql`
5. Click **Run** to execute the migration
6. Run the verification queries at the bottom to confirm success

Option 2: Node.js Migration Script

If you have a local PostgreSQL connection configured:

```
# From the project root directory
node scripts/run-migration-014.js
```

Option 3: Direct SQL Execution

Connect to your database using `psql` or another SQL client and run:

```
psql $DATABASE_URL -f database/migrations/014_fix_payment_id_column_type.sql
```

Verification

After applying the migration, verify it was successful by running:

```sql
-- Check column type
SELECT
  column_name,
  data_type
FROM information_schema.columns
WHERE table_name = 'credit_transactions'
AND column_name = 'payment_id';
```

Expected result: `data_type` should be **text**

Testing the Fix

1. Log in to the admin dashboard as a SUPER_ADMIN
2. Navigate to the **Customers** tab
3. Click **Add Credits** on any customer
4. Enter an amount (e.g., 50) and a description (e.g., "Test credit addition")
5. Click **Adding…** button
6. Verify that:
   - The operation completes successfully
   - No error message appears
   - The customer's credit balance updates correctly
   - The transaction appears in the audit logs

Additional Notes

About the 401 Errors

The screenshot also shows 401 (Unauthorized) errors on `/api/credits/balance`. These errors occur when:

1. The user's session has expired
2. The authentication token is invalid
3. The API endpoint is being called before authentication completes

Recommended fix: These should resolve automatically after a page refresh or re-login. If they persist, check:
- NextAuth session configuration
- Cookie settings
- CORS configuration

Payment ID Formats

After this migration, the `payment_id` column accepts the following formats:

- **Stripe payment intents:** `pi_xxxxxxxxxxxxx`

- **Stripe charges:** `ch_xxxxxxxxxxxxx`
- **Manual admin additions:** `MANUAL_ADMIN_{userId}_{timestamp}`
- **NULL:** For transactions not linked to payments

Files Modified

- `database/migrations/014_fix_payment_id_column_type.sql` - PostgreSQL migration
- `database/migrations/SUPABASE_MIGRATION_014.sql` - Supabase-specific migration
- `scripts/run-migration-014.js` - Migration runner script
- `ADMIN_CREDIT_FIX_GUIDE.md` - This guide

Rollback Instructions

If you need to rollback this migration (not recommended unless absolutely necessary):

```sql
BEGIN;

-- Drop the new constraint and index
ALTER TABLE credit_transactions
DROP CONSTRAINT IF EXISTS credit_transactions_payment_id_format_check;

DROP INDEX IF EXISTS idx_credit_transactions_payment_id;

-- Change payment_id back to INTEGER (this will fail if non-numeric IDs exist)
ALTER TABLE credit_transactions
ALTER COLUMN payment_id TYPE INTEGER USING payment_id::INTEGER;

-- Re-add the foreign key constraint
ALTER TABLE credit_transactions
ADD CONSTRAINT credit_transactions_payment_id_fkey
FOREIGN KEY (payment_id) REFERENCES stripe_payments(id) ON DELETE SET NULL;

COMMIT;
```

Warning: The rollback will fail if there are any non-numeric `payment_id` values in the database (including the manual admin IDs).

Support

If you encounter any issues:

1. Check the Supabase logs for detailed error messages
2. Verify that all previous migrations have been applied
3. Ensure you have sufficient database privileges to ALTER tables
4. Review the transaction history to ensure no data was corrupted

Next Steps

After successfully applying this migration:

1. ✅ Test adding credits through the admin dashboard
2. ✅ Verify credit balances update correctly

3. ✅ Check audit logs for proper transaction recording
4. ✅ Test Stripe payment processing to ensure it still works
5. ✅ Commit and push changes to your repository

Migration Date: December 1, 2025
Migration Number: 014
Status: Ready to apply