https://www.linkedin.com/in/praveenmadupu/
https://github.com/PMSQLDBA

Database Mail Configuration in SQL Server with Gmail SMTP Details (Step-by-step (SSMS + T-SQL))

Below are two complete ways to configure Database Mail (DBMail) in SQL Server to send mail through Gmail's SMTP:
1) GUI (SSMS Database Mail Configuration Wizard) and
2) scripted T-SQL (repeatable, automatable).

Also included testing, troubleshooting, security notes, and common pitfalls.

Important precondition:

- Google no longer allows plain username/password from apps unless you use an **App password** (when your Google account has 2-Step Verification enabled).
- Create an App password in your Google Account (Security → 2-Step Verification → App passwords) and use that 16-character password in place of your normal Google password.
- Do not attempt to enable "less secure apps" — it's deprecated.

Quick SMTP settings for Gmail

- SMTP server: smtp.gmail.com
- Port and encryption:
  - Port **587** with STARTTLS (TLS) — recommended
  - Or port **465** with SSL (less commonly used in DB mail)
- Authentication: required — use full Gmail address as username and an **App password** as password
- From address: your Gmail address (e.g. your.email@gmail.com)

A. GUI method (SSMS)

1. Open SQL Server Management Studio and connect to the instance.
2. In Object Explorer expand: Management → right-click Database Mail → **Configure Database Mail**.
3. If Database Mail is not enabled, Wizard will prompt — enable it.
4. Choose **Set up Database Mail by performing the following tasks** → Next.
5. Enter a **Profile name** (e.g. GmailProfile) and optional description. Click **Add** to create an account.
6. In the **New Database Mail Account** dialog set:
   - Account name: GmailAccount
   - Email address: your.email@gmail.com
   - Display name: e.g. SQL Server Alerts
   - Reply email: optional
   - Server name: smtp.gmail.com
   - Port number: 587
   - Check **This server requires a secure connection (SSL)** (for STARTTLS/SSL; SSMS labels may differ)
   - Authentication: select **Basic authentication** and supply Username: your.email@gmail.com and **Password: <app_password>**
7. Click OK. Back in profile screen, select newly created account and add to profile. Next.

8. Manage profile security: select whether profile is public (all DB mail users may use) or private (grant to specific SQL logins). Usually make it **public** for tests, but production best practice is **private** and then add specific principals. Click Next.
9. Configure system parameters (leave defaults usually fine): account retry attempts, retry delay, max file size, etc. Next and Finish.
10. After success, use the GUI to send a test email (right-click Database Mail → Send Test E-Mail) or use T-SQL below.

B. Scripted method (T-SQL)

Run these commands in msdb context.

Replace placeholders (your.email@gmail.com, <app_password>, profile/account names) before running.

```
USE msdb;
GO
-- 1. Enable Database Mail if disabled
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE WITH OVERRIDE;
EXEC sp_configure 'Database Mail XPs', 1;
RECONFIGURE WITH OVERRIDE;

-- 2. Create the Database Mail account
EXEC msdb.dbo.sysmail_add_account_sp
    @account_name = 'GmailAccount',
    @description = 'Gmail SMTP account for Database Mail',
    @email_address = 'your.email@gmail.com',
    @display_name = 'SQL Server Alerts',
    @replyto_address = 'your.email@gmail.com',
    @mailserver_name = 'smtp.gmail.com',
    @port = 587,
    @enable_ssl = 1,
    @username = 'your.email@gmail.com',
    @password = 'your_app_password_here';  -- use the Google App Password

-- 3. Create a Database Mail profile
EXEC msdb.dbo.sysmail_add_profile_sp
    @profile_name = 'GmailProfile',
    @description = 'Profile using Gmail SMTP';

-- 4. Add the account to the profile
EXEC msdb.dbo.sysmail_add_profileaccount_sp
    @profile_name = 'GmailProfile',
    @account_name = 'GmailAccount',
    @sequence_number = 1;
```

https://www.linkedin.com/in/praveenmadupu/
https://github.com/PMSQLDBA

```
-- 5. (Optional) Make the profile public so all logins can use it
        EXEC msdb.dbo.sysmail_add_principalprofile_sp
          @profile_name = 'GmailProfile',
          @principal_name = 'public',
          @is_default = 1;

-- 6. Send a test email
        EXEC msdb.dbo.sp_send_dbmail
          @profile_name = 'GmailProfile',
          @recipients = 'recipient@example.com',
          @subject = 'DBMail Gmail test',
          @body = 'This is a test message sent via Gmail SMTP.';
```

C. Test and confirm delivery

- After running sp_send_dbmail, check Database Mail Log in SSMS: Object Explorer → Management → Database Mail → View Database Mail Log.
- Query diagnostic tables for details:

```
-- events
SELECT TOP 50 * FROM msdb.dbo.sysmail_event_log ORDER BY log_date DESC;

-- sent/failed items
SELECT * FROM msdb.dbo.sysmail_sentitems ORDER BY send_request_date DESC;
SELECT * FROM msdb.dbo.sysmail_faileditems ORDER BY last_mod_date DESC;
```


D. Troubleshooting (common failures)

1. **Authentication error / login failed**
   - Ensure you used a Google **App password** and not your regular password.
   - Ensure the Gmail account has 2-Step Verification enabled and the App password created for “Mail” or “Other (custom)”.
2. **Connection timed out / cannot connect to smtp.gmail.com**
   - Verify outbound firewall allows the SMTP port (587). Many corporate/cloud environments block outbound SMTP.
   - From the SQL Server host, test connectivity: telnet smtp.gmail.com 587 or openssl s_client -starttls smtp -crlf -connect smtp.gmail.com:587 (if tools available).
3. **SSL/TLS handshake or certificate errors**
   - Ensure @enable_ssl = 1. Port 587 uses STARTTLS; some older SQL Server versions/OS stacks may not support newer TLS versions—ensure Windows/SQL Server have up-to-date TLS support.
4. **Messages stuck in queue**
   - Check sysmail_faileditems and sysmail_unsentitems. Examine sysmail_event_log for error messages.
   - Restart the Database Mail external program (SQL Server Agent not required) or restart the SQL Server instance if stuck (as last resort).

5. **Gmail blocking sign-in attempt**
    - Google may block login attempts it perceives as risky. Use App passwords and, if necessary, check the Google Account Security → Recent security events / blocked sign-in attempts and allow.

E. Security & operational best practices

- Use **App passwords**, not account main password. Rotate app passwords periodically.
- Prefer making the profile **private** and assign it only to necessary msdb principals. Use sysmail_add_principalprofile_sp to assign specific logins/roles.
- Monitor sysmail_event_log and set up alerts for failed mails.
- Consider using a dedicated service account email rather than a personal Gmail address for production.
- For high volume or enterprise scenarios, consider using a corporate SMTP relay or transactional email service (SendGrid, Amazon SES, etc.) instead of Gmail.

🚨 **Gmail will NOT authenticate unless you use an App Password**

- SQL Server Database Mail **cannot** use Gmail's normal password.
- Google permanently removed "Less Secure Apps" support.

Step 1 — Enable 2-Step Verification

Go to: https://myaccount.google.com/security

Turn on:

✓ **2-Step Verification** → You must turn this on before Google allows App Passwords.

Step 2 — Create a Gmail App Password

Go to:

https://myaccount.google.com/apppasswords

Log in → choose **Mail** and **Windows Computer** → Generate.

Google will give you a **16-character password** like:

abcd efgh ijkl mnop

This is the ONLY password Gmail will accept for SMTP.

Step 3 — Update Database Mail Account

In **SSMS**:

Management → Database Mail → Manage Accounts → Edit Account

SMTP settings:

- **Server**: smtp.gmail.com
- **Port**: 587
- **Enable SSL**: ✓ **checked**

Authentication:

- **Basic Authentication**
- **Username**: your full Gmail address
- **Password**: the **App Password**, not your Gmail password

Save changes.

https://www.linkedin.com/in/praveenmadupu/
https://github.com/PMSQLDBA

Step 4 — Restart Mail Queue

```
EXEC msdb.dbo.sysmail_stop_sp;
EXEC msdb.dbo.sysmail_start_sp;
```

Step 5 — Test

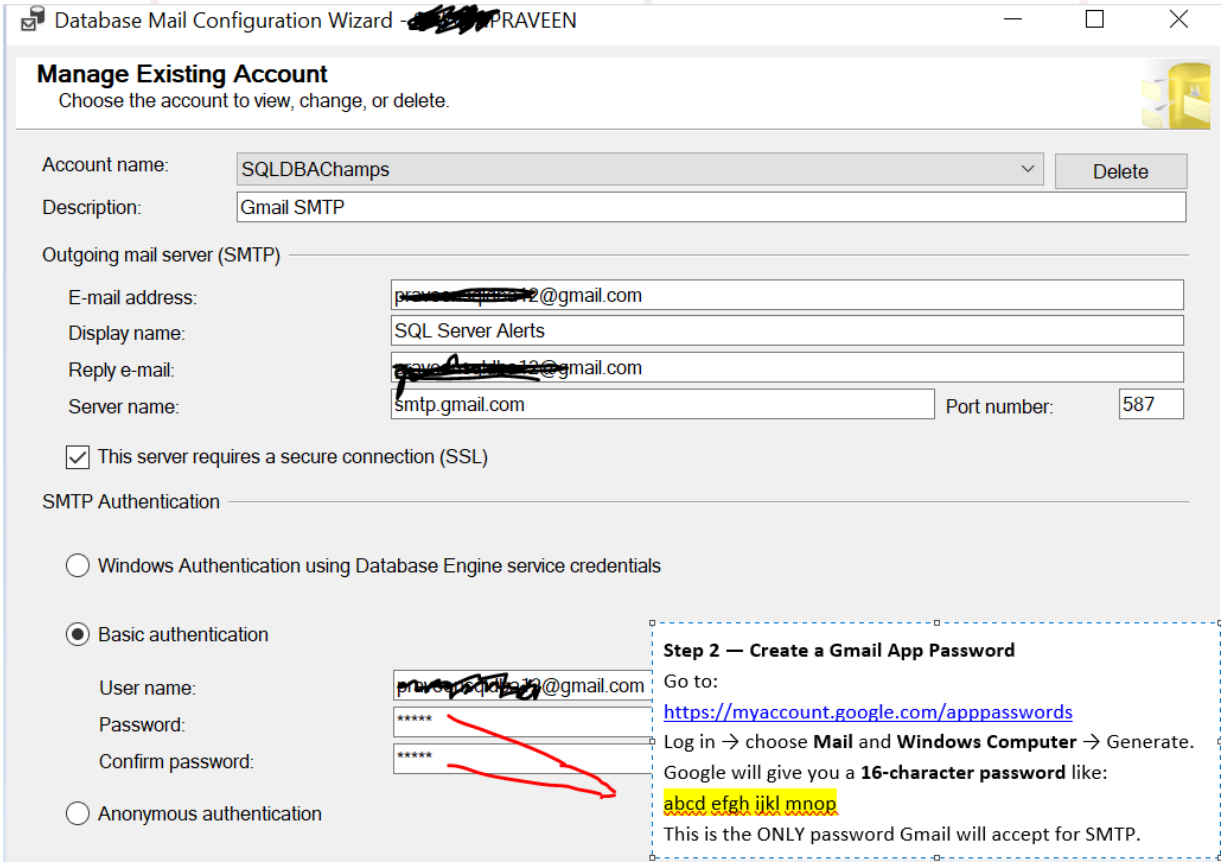
```
EXEC msdb.dbo.sp_send_dbmail
   @profile_name = 'YourProfileName',
   @recipients = 'youraddress@gmail.com',
   @subject = 'Test From SQL Server',
   @body = 'Gmail + SQL Server Mail working!';
```

🎉 **Your Gmail setup will now succeed**

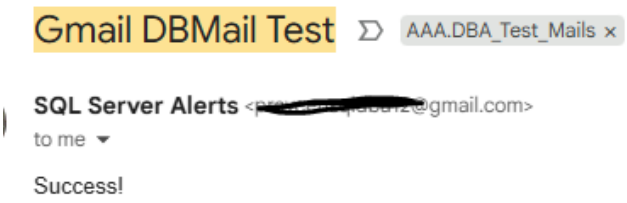
This setup is the **only** working and supported way to send email from SQL Server using Gmail in 2025.

https://www.linkedin.com/in/praveenmadupu/
https://github.com/PMSQLDBA

```
USE msdb;
GO

EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'Database Mail XPs', 1;
RECONFIGURE;
GO

-- Create account
EXEC sysmail_add_account_sp
   @account_name = 'SQLDBAChamps',
   @description = 'Gmail SMTP',
   @email_address = 'DBAAlerts2025@gmail.com',
   @display_name = 'SQL Server Alerts',
   @mailserver_name = 'smtp.gmail.com',
   @port = 587,            -- required for STARTTLS
   @enable_ssl = 1,        -- **critical**
   @username = 'DBAAlerts2025@gmail.com',
   @password = 'Welcome@12345$';
-- 16-char app password that you will get form your gmail account as shown in below picture.
GO
```

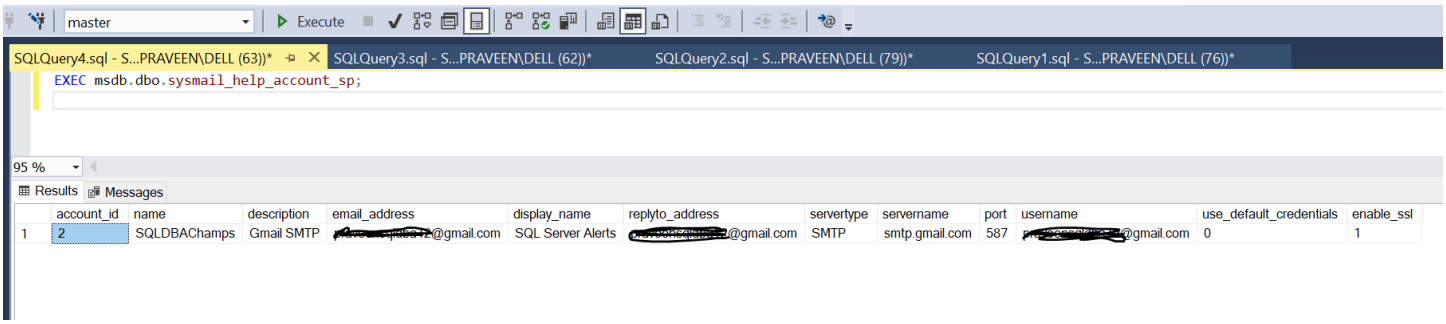
```
-- Create profile
EXEC sysmail_add_profile_sp
   @profile_name = 'DBAAlerts',
   @description = 'SMTP profile';
GO

-- Bind account to profile
EXEC sysmail_add_profileaccount_sp
   @profile_name = 'DBAAlerts',
   @account_name = 'SQLDBAChamps',
   @sequence_number = 1;
GO

-- Make default (optional)
EXEC sysmail_add_principalprofile_sp
   @profile_name = 'DBAAlerts',
   @principal_name = 'public',
   @is_default = 1;
GO

EXEC msdb.dbo.sp_send_dbmail
   @profile_name = 'DBAAlerts',
   @recipients = 'DBAAlerts2025@gmail.com',
   @subject = 'Gmail DBMail Test',
   @body = 'Success!';
```

https://www.linkedin.com/in/praveenmadupu/
https://github.com/PMSQLDBA

Verification Scripts:

```
EXEC msdb.dbo.sysmail_help_account_sp;
```



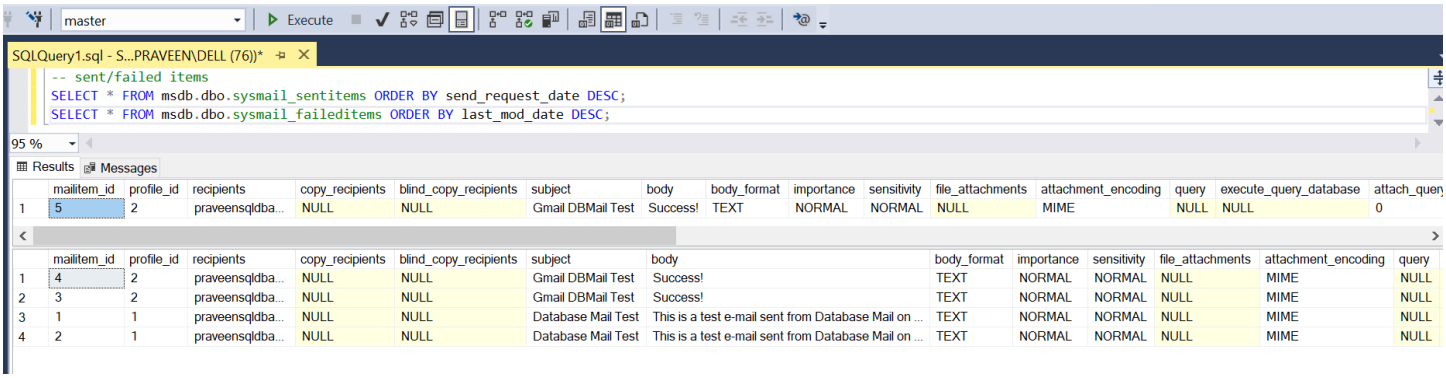
	account_id	name	description	email_address	display_name	replyto_address	servertype	servername	port	username	use_default_credentials	enable_ssl
1	2	SQLDBAChamps	Gmail SMTP	[redacted]@gmail.com	SQL Server Alerts	[redacted]@gmail.com	SMTP	smtp.gmail.com	587	[redacted]@gmail.com	0	1

-- events

```
SELECT TOP 10 * FROM msdb.dbo.sysmail_event_log ORDER BY log_date DESC;
```

-- sent/failed items

```
SELECT * FROM msdb.dbo.sysmail_sentitems ORDER BY send_request_date DESC;
SELECT * FROM msdb.dbo.sysmail_faileditems ORDER BY last_mod_date DESC;
```



	mailitem_id	profile_id	recipients	copy_recipients	blind_copy_recipients	subject	body	body_format	importance	sensitivity	file_attachments	attachment_encoding	query	execute_query_database	attach_query
1	5	2	praveensqldba...	NULL	NULL	Gmail DBMail Test	Success!	TEXT	NORMAL	NORMAL	NULL	MIME	NULL	NULL	0

	mailitem_id	profile_id	recipients	copy_recipients	blind_copy_recipients	subject	body	body_format	importance	sensitivity	file_attachments	attachment_encoding	query
1	4	2	praveensqldba...	NULL	NULL	Gmail DBMail Test	Success!	TEXT	NORMAL	NORMAL	NULL	MIME	NULL
2	3	2	praveensqldba...	NULL	NULL	Gmail DBMail Test	Success!	TEXT	NORMAL	NORMAL	NULL	MIME	NULL
3	1	1	praveensqldba...	NULL	NULL	Database Mail Test	This is a test e-mail sent from Database Mail on ...	TEXT	NORMAL	NORMAL	NULL	MIME	NULL
4	2	1	praveensqldba...	NULL	NULL	Database Mail Test	This is a test e-mail sent from Database Mail on ...	TEXT	NORMAL	NORMAL	NULL	MIME	NULL