FULL SCHEMA ANALYSIS SCRIPT — single .sql file

```
/* ============================================================
  SQL SERVER SCHEMA ANALYSIS BUNDLE
  Complete one-file script for structural + data-quality checks
  ============================================================ */



/* ------------------------------------------------------------
  1. List all tables, columns, and data types
  ------------------------------------------------------------ */
PRINT '1. TABLE + COLUMN INVENTORY';
SELECT
   t.TABLE_SCHEMA,
   t.TABLE_NAME,
   c.COLUMN_NAME,
   c.DATA_TYPE,
   c.IS_NULLABLE,
   c.CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.TABLES t
JOIN INFORMATION_SCHEMA.COLUMNS c
   ON t.TABLE_NAME = c.TABLE_NAME
   AND t.TABLE_SCHEMA = c.TABLE_SCHEMA
WHERE t.TABLE_TYPE = 'BASE TABLE'
ORDER BY t.TABLE_SCHEMA, t.TABLE_NAME, c.ORDINAL_POSITION;
```

```
/* ------------------------------------------------------------
   2. Primary Keys (Existing + Missing)
   ------------------------------------------------------------ */
PRINT '2. PRIMARY KEYS — EXISTING';
SELECT
    t.name AS TableName,
    i.name AS PrimaryKeyName,
    COL_NAME(ic.object_id, ic.column_id) AS ColumnName
FROM sys.tables t
JOIN sys.indexes i ON t.object_id = i.object_id AND i.is_primary_key = 1
JOIN sys.index_columns ic ON i.object_id = ic.object_id AND i.index_id = ic.index_id
ORDER BY t.name;

PRINT '2. PRIMARY KEYS — MISSING';
SELECT t.name AS TableWithoutPK
FROM sys.tables t
LEFT JOIN sys.indexes i
    ON t.object_id = i.object_id
    AND i.is_primary_key = 1
WHERE i.object_id IS NULL;


/* ------------------------------------------------------------
   3. Foreign Keys + Orphan Analysis
```

```
  ------------------------------------------------------------ */
PRINT '3. FOREIGN KEYS';
SELECT
   fk.name AS ForeignKey,
   OBJECT_NAME(fk.parent_object_id) AS ChildTable,
   cpa.name AS ChildColumn,
   OBJECT_NAME(fk.referenced_object_id) AS ParentTable,
   cref.name AS ParentColumn
FROM sys.foreign_keys fk
JOIN sys.foreign_key_columns fkc ON fk.object_id = fkc.constraint_object_id
JOIN sys.columns cpa ON cpa.column_id = fkc.parent_column_id AND cpa.object_id =
fk.parent_object_id
JOIN sys.columns cref ON cref.column_id = fkc.referenced_column_id AND cref.object_id =
fk.referenced_object_id
ORDER BY ParentTable, ChildTable;

PRINT '3. ORPHAN RECORD DETECTION (per FK)';
DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql += '
SELECT ''' + fk.name + ''' AS ForeignKeyName,
    ''' + OBJECT_NAME(fk.parent_object_id) + ''' AS ChildTable,
    *
FROM ' + QUOTENAME(OBJECT_SCHEMA_NAME(fk.parent_object_id)) + '.' +
QUOTENAME(OBJECT_NAME(fk.parent_object_id)) + ' c
LEFT JOIN ' + QUOTENAME(OBJECT_SCHEMA_NAME(fk.referenced_object_id)) + '.' +
QUOTENAME(OBJECT_NAME(fk.referenced_object_id)) + ' p
```

```
ON c.' + cpa.name + ' = p.' + cref.name + '

WHERE p.' + cref.name + ' IS NULL;

'

FROM sys.foreign_keys fk

JOIN sys.foreign_key_columns fkc ON fk.object_id = fkc.constraint_object_id

JOIN sys.columns cpa ON cpa.column_id = fkc.parent_column_id AND cpa.object_id =
fk.parent_object_id

JOIN sys.columns cref ON cref.column_id = fkc.referenced_column_id AND cref.object_id =
fk.referenced_object_id;


EXEC sp_executesql @sql;




/* ------------------------------------------------------------

  4. Missing Indexes

  ------------------------------------------------------------ */

PRINT '4. MISSING INDEXES (ranked by impact)';

SELECT

   mid.database_id,

   OBJECT_NAME(mid.object_id) AS TableName,

   mid.equality_columns,

   mid.inequality_columns,

   mid.included_columns,

   migs.user_seeks + migs.user_scans AS TotalReads

FROM sys.dm_db_missing_index_details mid

JOIN sys.dm_db_missing_index_groups mig ON mid.index_handle = mig.index_handle
```

```
JOIN sys.dm_db_missing_index_group_stats migs ON mig.index_group_handle =
migs.group_handle

ORDER BY TotalReads DESC;




/* -----------------------------------------------------------

  5. Unused & Duplicate Indexes

  ----------------------------------------------------------- */

PRINT '5. UNUSED INDEXES';

SELECT

   OBJECT_NAME(i.object_id) AS TableName,

   i.name AS IndexName,

   i.index_id,

   user_seeks, user_scans, user_lookups, user_updates

FROM sys.indexes i

LEFT OUTER JOIN sys.dm_db_index_usage_stats s

   ON i.object_id = s.object_id AND i.index_id = s.index_id

WHERE OBJECTPROPERTY(i.object_id, 'IsUserTable') = 1

AND (user_seeks IS NULL AND user_scans IS NULL AND user_lookups IS NULL)

ORDER BY i.object_id;


PRINT '5. DUPLICATE INDEXES';

SELECT

   OBJECT_NAME(ic1.object_id) AS TableName,

   i1.name AS Index1,
```

```
    i2.name AS Index2,
    c1.name AS ColumnName
FROM sys.index_columns ic1
JOIN sys.index_columns ic2
    ON ic1.object_id = ic2.object_id
    AND ic1.column_id = ic2.column_id
JOIN sys.indexes i1 ON ic1.object_id = i1.object_id AND ic1.index_id = i1.index_id
JOIN sys.indexes i2 ON ic2.object_id = i2.object_id AND ic2.index_id = i2.index_id
JOIN sys.columns c1 ON ic1.object_id = c1.object_id AND ic1.column_id = c1.column_id
WHERE i1.index_id < i2.index_id;



/* ------------------------------------------------------------
   6. NULL PATTERN ANALYSIS
   ------------------------------------------------------------ */
PRINT '6. NULL ANALYSIS — COLUMN NULL COUNT PER TABLE';

DECLARE @table NVARCHAR(256), @q NVARCHAR(MAX);

DECLARE tbl CURSOR FOR
SELECT QUOTENAME(TABLE_SCHEMA) + '.' + QUOTENAME(TABLE_NAME)
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE';

OPEN tbl;
```

```
FETCH NEXT FROM tbl INTO @table;

WHILE @@FETCH_STATUS = 0
BEGIN
   SET @q = '
      SELECT ''' + @table + ''' AS TableName, COLUMN_NAME,
            (SELECT COUNT(*) FROM ' + @table + ') AS TotalRows,
            (SELECT COUNT(*) FROM ' + @table + ' WHERE ' + COLUMN_NAME + ' IS NULL) AS
NullCount
      FROM INFORMATION_SCHEMA.COLUMNS
      WHERE TABLE_SCHEMA + ''.'' + TABLE_NAME = ''' + REPLACE(@table, '[', '') + '''
          AND TABLE_SCHEMA + ''.'' + TABLE_NAME = ''' + REPLACE(@table, ']', '') + ''';
   ';
   EXEC (@q);
   FETCH NEXT FROM tbl INTO @table;
END
CLOSE tbl;
DEALLOCATE tbl;



/* ------------------------------------------------------------
  7. Duplicate Detection (template)
  ------------------------------------------------------------ */
PRINT '7. DUPLICATE KEY DETECTION TEMPLATE';
PRINT 'Modify YourTable and KeyColumn before running:';
```

```
-- Example:
-- SELECT KeyColumn, COUNT(*)
-- FROM YourTable
-- GROUP BY KeyColumn
-- HAVING COUNT(*) > 1;



/* ------------------------------------------------------------
   8. Detect VARCHAR columns storing numeric or date data
   ------------------------------------------------------------ */
PRINT '8. VARCHAR COLUMNS SUSPECTED OF STORING DATES/NUMBERS';
SELECT
    TABLE_NAME, COLUMN_NAME
FROM INFORMATION_SCHEMA.COLUMNS
WHERE DATA_TYPE IN ('varchar', 'nvarchar')
AND (COLUMN_NAME LIKE '%date%' OR COLUMN_NAME LIKE '%num%' OR COLUMN_NAME
LIKE '%id%');



/* ------------------------------------------------------------
   9. Table Row Counts (large/small tables)
   ------------------------------------------------------------ */
PRINT '9. ROW COUNTS PER TABLE';
SELECT
```

```
    s.name AS SchemaName,
    t.name AS TableName,
    p.rows AS RowCount
FROM sys.tables t
JOIN sys.schemas s ON t.schema_id = s.schema_id
JOIN sys.partitions p ON t.object_id = p.object_id AND p.index_id IN (0, 1)
ORDER BY RowCount DESC;




/* ------------------------------------------------------------
   10. Unused / Orphaned Tables
   ------------------------------------------------------------ */
PRINT '10. UNUSED/ORPHANED TABLES';
SELECT
    t.name AS TableName,
    MAX(ps.last_user_scan) AS LastScan,
    MAX(ps.last_user_seek) AS LastSeek,
    MAX(ps.last_user_update) AS LastUpdate
FROM sys.tables t
LEFT JOIN sys.dm_db_index_usage_stats ps ON t.object_id = ps.object_id
GROUP BY t.name
ORDER BY LastSeek, LastScan, LastUpdate;
```