

Data Dictionary

Data Dictionary

2022-03-21

TRIAL

TRIAL

TRIAL

## Table of contents

1. Sample Business Glossary .....	18
1.1. Business Glossary .....	19
1.1. Terms .....	19
1.1.1. Term: Bill Of Materials .....	19
1.1.2. Term: Billing Address .....	20
1.1.3. Term: Company .....	21
1.1.4. Term: Company name .....	22
1.1.5. Term: Customer .....	23
1.1.6. Term: Customer Product Review .....	24
1.1.7. Term: Customer Purchase Order Number .....	25
1.1.8. Term: Department .....	26
1.1.9. Term: Freight .....	27
1.1.10. Term: Inventory Location .....	28
1.1.11. Term: Manufacturing Location .....	29
1.1.12. Term: Mfg Location Capacity .....	30
1.1.13. Term: Mfg Location Cost Rate .....	31
1.1.14. Term: Online (Sales) Order .....	32
1.1.15. Term: Product .....	33
1.1.16. Term: Product Category .....	34
1.1.17. Term: Product Model .....	35
1.1.18. Term: Product Number .....	36
1.1.19. Term: Product Standard Cost .....	37
1.1.20. Term: Purchase Order .....	38
1.1.21. Term: Purchase Order Date .....	39
1.1.22. Term: Purchase Order Due Date .....	40
1.1.23. Term: Purchase Order Quantity .....	41
1.1.24. Term: Purchase Order Received Quantity .....	42
1.1.25. Term: Purchase Order Rejected Quantity .....	43
1.1.26. Term: Purchase Order Ship Date .....	44
1.1.27. Term: Purchase Order Stocked Quantity .....	45
1.1.28. Term: Sales Bonus .....	46
1.1.29. Term: Sales Commission .....	47
1.1.30. Term: Sales Order .....	48
1.1.31. Term: Sales Order Date .....	49
1.1.32. Term: Sales Order Due Date .....	50
1.1.33. Term: Sales Order Number .....	51
1.1.34. Term: Sales Order Ship Date .....	52
1.1.35. Term: Sales Person .....	53
1.1.36. Term: Sales Person Quota .....	54
1.1.37. Term: Sales Person Sales .....	55
1.1.38. Term: Sales Reason .....	56
1.1.39. Term: Sales Territory .....	57
1.1.40. Term: Shift .....	58
1.1.41. Term: Ship Base .....	59
1.1.42. Term: Ship Method .....	60

1.1.43. Term: Ship Rate .....	61
1.1.44. Term: Shipping Address .....	62
1.1.45. Term: Shopping Cart .....	63
1.1.46. Term: Store .....	64
1.1.47. Term: Ticker symbol .....	65
1.1.48. Term: Vendor .....	66
1.1.49. Term: Vendor Credit Rating .....	67
1.1.50. Term: Warehouse .....	68
1.1.51. Term: Work Order .....	69
1.2. Categories .....	70
1.2.1. Category: Inventory .....	70
1.2.2. Category: Manufacturing .....	71
1.2.3. Category: Procurement .....	72
1.2.4. Category: Sales .....	73
2. ExamSystem .....	74
2.1. Data Dictionary .....	75
2.1.1. Tables .....	75
2.1.1.1. Table: Choice .....	75
2.1.1.2. Table: Course .....	76
2.1.1.3. Table: Crs_Top .....	77
2.1.1.4. Table: Department .....	78
2.1.1.5. Table: Exam .....	79
2.1.1.6. Table: Exm_Ques .....	80
2.1.1.7. Table: Ins_Crs .....	81
2.1.1.8. Table: Instructor .....	82
2.1.1.9. Table: Question .....	84
2.1.1.10. Table: Std_Crs .....	86
2.1.1.11. Table: Std_Ques_Exm .....	87
2.1.1.12. Table: Student .....	88
2.1.1.13. Table: User .....	90
2.1.2. Procedures .....	91
2.1.2.1. Procedure: DelCrsIns .....	91
2.1.2.2. Procedure: DeleteInstructor .....	92
2.1.2.3. Procedure: DeleteStdByld .....	93
2.1.2.4. Procedure: DeleteUserByUserName .....	94
2.1.2.5. Procedure: InsertInsForCrs .....	95
2.1.2.6. Procedure: InsertInstructor .....	96
2.1.2.7. Procedure: InsertNewStd .....	97
2.1.2.8. Procedure: InsertNewUser .....	98
2.1.2.9. Procedure: InsertQues .....	99
2.1.2.10. Procedure: SelectCrsIns .....	100
2.1.2.11. Procedure: SelectInstructor .....	101
2.1.2.12. Procedure: SelectStdByld .....	102
2.1.2.13. Procedure: SelectUserByld .....	103
2.1.2.14. Procedure: sp_AssignAnsToStd .....	104
2.1.2.15. Procedure: sp_CorrectStdExam .....	105

2.1.2.16. Procedure: sp_CourseNameWithStudentCount .....	107
2.1.2.17. Procedure: sp_deleteChoice .....	108
2.1.2.18. Procedure: sp_deleteCourse .....	109
2.1.2.19. Procedure: sp_deleteDepartment .....	110
2.1.2.20. Procedure: sp_deleteExam .....	111
2.1.2.21. Procedure: sp_DeleteQues .....	112
2.1.2.22. Procedure: sp_DeleteStdFromCrs .....	113
2.1.2.23. Procedure: sp_deleteTopic .....	114
2.1.2.24. Procedure: sp_ExamQuestionsAndAnswers .....	115
2.1.2.25. Procedure: sp_GenerateExam .....	116
2.1.2.26. Procedure: sp_InsertChoice .....	117
2.1.2.27. Procedure: sp_insertCourse .....	118
2.1.2.28. Procedure: sp_insertDepartment .....	119
2.1.2.29. Procedure: sp_insertExam .....	120
2.1.2.30. Procedure: sp_InsertStdAnsForExm .....	121
2.1.2.31. Procedure: sp_InsertStdForCrs .....	122
2.1.2.32. Procedure: sp_insertTopic .....	123
2.1.2.33. Procedure: sp_selectchoice .....	124
2.1.2.34. Procedure: sp_selectCourse .....	125
2.1.2.35. Procedure: sp_selectDepartment .....	126
2.1.2.36. Procedure: sp_selectExam .....	127
2.1.2.37. Procedure: sp_SelectQues .....	128
2.1.2.38. Procedure: sp_SelectStdAnsForExm .....	129
2.1.2.39. Procedure: sp_SelectStdOfCrs .....	130
2.1.2.40. Procedure: sp_selectTopic .....	131
2.1.2.41. Procedure: sp_StdExamAnswers .....	132
2.1.2.42. Procedure: sp_StudentGradePerCourse .....	133
2.1.2.43. Procedure: sp_StudentperDepartment .....	134
2.1.2.44. Procedure: sp_TopicsPerCourse .....	135
2.1.2.45. Procedure: sp_updateChoice .....	136
2.1.2.46. Procedure: sp_updateCourse .....	137
2.1.2.47. Procedure: sp_UpdateCrsForIns .....	138
2.1.2.48. Procedure: sp_UpdateCrsOfStd .....	139
2.1.2.49. Procedure: sp_updateDepartment .....	140
2.1.2.50. Procedure: sp_updateExam .....	141
2.1.2.51. Procedure: sp_updateTopic .....	142
2.1.2.52. Procedure: UpdateInstructor .....	143
2.1.2.53. Procedure: UpdateQues .....	144
2.1.2.54. Procedure: UpdateStdInfo .....	145
2.1.2.55. Procedure: UpdateUser .....	146
3. Sample Data Lake .....	148
3.1. US Census - 2018 Business Dynamics .....	149
3.1.1. Structures .....	149
3.1.1.1. Structure: bds2018_cty.csv (Business Dynamics 2018 - City) .....	149
3.1.1.2. Structure: bds2018_eage.csv (Business Dynamics 2018 - Establishment Age) .....	151
3.1.1.3. Structure: bds2018_esize.csv (Business Dynamics 2018 - Establishment Size) .....	153

3.1.1.4. Structure: bds2018_fage.csv (Business Dynamics 2018 - Firm Age) .....	155
3.1.1.5. Structure: bds2018_fsize.csv (Business Dynamics 2018 - Firm Size) .....	157
3.1.1.6. Structure: bds2018_iesize.csv (Business Dynamics 2018 - Initial Establishment Size) .....	159
3.1.1.7. Structure: bds2018_ifsize.csv (Business Dynamics 2018 - Initial Firm Size) .....	161
3.1.1.8. Structure: bds2018_metro.csv (Business Dynamics 2018 - Metro/Non-Metro) .....	163
3.1.1.9. Structure: bds2018_sector.csv (Business Dynamics 2018 - Sector) .....	165
3.1.1.10. Structure: bds2018_st.csv (Business Dynamics 2018 - State) .....	167
3.1.1.11. Structure: bds2018.csv (Business Dynamics 2018 - Economy-wide) .....	169
3.2. Fortune 500 .....	172
3.2.1. Structures .....	172
3.2.1.1. Structure: 2015Fortune500.csv (Fortune 500 Companies (2015)) .....	172
3.2.1.2. Structure: Fortune500.csv (Fortune 500 Companies (2016)) .....	173
3.3. Geography .....	175
3.3.1. Structures .....	175
3.3.1.1. Structure: zips.json (Zips) .....	175
3.3.1.2. Structure: country.xml (Countries) .....	176
3.4. U.S. Securities and Exchange Commission .....	179
3.4.1. Structures .....	179
3.4.1.1. Structure: company_tickers.json (Company tickers) .....	179
3.5. HDFS formats .....	181
3.5.1. Structures .....	181
3.5.1.1. Structure: users.avro.[example.avro.User] .....	181
3.5.1.2. Structure: users.orc .....	182
3.5.1.3. Structure: users.parquet .....	183
3.6. Insurance (Excel) .....	185
3.6.1. Structures .....	185
3.6.1.1. Structure: insurance_data.xlsx.tblPolicies (Insurance Data) .....	185
3.7. Employees (Delta Lake) .....	187
3.7.1. Structures .....	187
3.7.1.1. Structure: employees.Delta (Employees) .....	187
4. Sample Data Profiling Database .....	189
4.1. Data Dictionary .....	190
4.1.1. Tables .....	190
4.1.1.1. Table: Clients (Clients) .....	190
4.1.1.2. Table: Items (Items in stock) .....	191
4.1.1.3. Table: Orders (Placed orders) .....	192
5. Sample Elasticsearch Database (Search Engine) .....	194
5.1. Data Dictionary .....	195
5.1.1. Tables .....	195
5.1.1.1. Search index: bank .....	195
5.1.1.2. Search index: kibana_sample_data_ecommerce .....	196
5.1.1.3. Search index: kibana_sample_data_flights .....	198
5.1.1.4. Search index: kibana_sample_data_logs .....	199
6. Sample MongoDB Database (NoSQL Document Store) .....	201
6.1. MongoDB schema .....	202
6.1.1. Tables .....	203

6.1.1.1. Collection: people .....	203
6.1.1.2. Collection: programs .....	204
6.1.1.3. Collection: projects .....	205
7. Sample Neo4j Database (NoSQL Graph Database) .....	208
7.1. Movies .....	209
7.1.1. Tables .....	210
7.1.1.1. Graph edge table: ACTED_IN .....	210
7.1.1.2. Graph edge table: DIRECTED .....	211
7.1.1.3. Graph edge table: FOLLOWS .....	212
7.1.1.4. Graph node table: Movie .....	213
7.1.1.5. Graph node table: Person .....	214
7.1.1.6. Graph edge table: PRODUCED .....	215
7.1.1.7. Graph edge table: REVIEWED .....	216
7.1.1.8. Graph edge table: WROTE .....	217
7.1.2. Other .....	219
7.2. Procedures .....	219
7.2.1.1. Procedure: db.awaitIndex .....	219
7.2.1.2. Procedure: db.awaitIndexes .....	220
7.2.1.3. Procedure: db.checkpoint .....	221
7.2.1.4. Procedure: db.clearQueryCaches .....	222
7.2.1.5. Procedure: db.constraints .....	223
7.2.1.6. Procedure: db.createIndex .....	224
7.2.1.7. Procedure: db.createLabel .....	225
7.2.1.8. Procedure: db.createNodeKey .....	226
7.2.1.9. Procedure: db.createProperty .....	227
7.2.1.10. Procedure: db.createRelationshipType .....	228
7.2.1.11. Procedure: db.createUniquePropertyConstraint .....	229
7.2.1.12. Procedure: db.index.fulltext.awaitEventuallyConsistentIndexRefresh .....	230
7.2.1.13. Procedure: db.index.fulltext.createNodeIndex .....	231
7.2.1.14. Procedure: db.index.fulltext.createRelationshipIndex .....	232
7.2.1.15. Procedure: db.index.fulltext.drop .....	233
7.2.1.16. Procedure: db.index.fulltext.listAvailableAnalyzers .....	234
7.2.1.17. Procedure: db.index.fulltext.queryNodes .....	235
7.2.1.18. Procedure: db.index.fulltext.queryRelationships .....	236
7.2.1.19. Procedure: db.indexDetails .....	237
7.2.1.20. Procedure: db.indexes .....	238
7.2.1.21. Procedure: db.info .....	239
7.2.1.22. Procedure: db.labels .....	240
7.2.1.23. Procedure: db.listLocks .....	241
7.2.1.24. Procedure: db.ping .....	242
7.2.1.25. Procedure: db.prepareForReplanning .....	243
7.2.1.26. Procedure: db.propertyKeys .....	244
7.2.1.27. Procedure: db.relationshipTypes .....	245
7.2.1.28. Procedure: db.resampleIndex .....	246
7.2.1.29. Procedure: db.resampleOutdatedIndexes .....	247
7.2.1.30. Procedure: db.schema.nodeTypeProperties .....	248

7.2.1.31. Procedure: db.schema.relTypeProperties .....	249
7.2.1.32. Procedure: db.schema.visualization .....	250
7.2.1.33. Procedure: db.schemaStatements .....	251
7.2.1.34. Procedure: db.stats.clear .....	252
7.2.1.35. Procedure: db.stats.collect .....	253
7.2.1.36. Procedure: db.stats.retrieve .....	254
7.2.1.37. Procedure: db.stats.retrieveAllAnonymized .....	255
7.2.1.38. Procedure: db.stats.status .....	256
7.2.1.39. Procedure: db.stats.stop .....	257
7.2.1.40. Procedure: dbms.cluster.overview .....	258
7.2.1.41. Procedure: dbms.cluster.protocols .....	259
7.2.1.42. Procedure: dbms.cluster.role .....	260
7.2.1.43. Procedure: dbms.cluster.routing.getRoutingTable .....	261
7.2.1.44. Procedure: dbms.components .....	262
7.2.1.45. Procedure: dbms.database.state .....	263
7.2.1.46. Procedure: dbms.functions .....	264
7.2.1.47. Procedure: dbms.info .....	265
7.2.1.48. Procedure: dbms.killConnection .....	266
7.2.1.49. Procedure: dbms.killConnections .....	267
7.2.1.50. Procedure: dbms.killQueries .....	268
7.2.1.51. Procedure: dbms.killQuery .....	269
7.2.1.52. Procedure: dbms.killTransaction .....	270
7.2.1.53. Procedure: dbms.killTransactions .....	271
7.2.1.54. Procedure: dbms.listActiveLocks .....	272
7.2.1.55. Procedure: dbms.listCapabilities .....	273
7.2.1.56. Procedure: dbms.listConfig .....	274
7.2.1.57. Procedure: dbms.listConnections .....	275
7.2.1.58. Procedure: dbms.listPools .....	276
7.2.1.59. Procedure: dbms.listQueries .....	277
7.2.1.60. Procedure: dbms.listTransactions .....	278
7.2.1.61. Procedure: dbms.procedures .....	279
7.2.1.62. Procedure: dbms.quarantineDatabase .....	280
7.2.1.63. Procedure: dbms.queryJmx .....	281
7.2.1.64. Procedure: dbms.routing.getRoutingTable .....	282
7.2.1.65. Procedure: dbms.scheduler.failedJobs .....	283
7.2.1.66. Procedure: dbms.scheduler.groups .....	284
7.2.1.67. Procedure: dbms.scheduler.jobs .....	285
7.2.1.68. Procedure: dbms.scheduler.profile .....	286
7.2.1.69. Procedure: dbms.security.activateUser .....	287
7.2.1.70. Procedure: dbms.security.addRoleToUser .....	288
7.2.1.71. Procedure: dbms.security.changePassword .....	289
7.2.1.72. Procedure: dbms.security.changeUserPassword .....	290
7.2.1.73. Procedure: dbms.security.clearAuthCache .....	291
7.2.1.74. Procedure: dbms.security.createRole .....	292
7.2.1.75. Procedure: dbms.security.createUser .....	293
7.2.1.76. Procedure: dbms.security.deleteRole .....	294

7.2.1.77. Procedure: dbms.security.deleteUser .....	295
7.2.1.78. Procedure: dbms.security.listRoles .....	296
7.2.1.79. Procedure: dbms.security.listRolesForUser .....	297
7.2.1.80. Procedure: dbms.security.listUsers .....	298
7.2.1.81. Procedure: dbms.security.listUsersForRole .....	299
7.2.1.82. Procedure: dbms.security.removeRoleFromUser .....	300
7.2.1.83. Procedure: dbms.security.suspendUser .....	301
7.2.1.84. Procedure: dbms.setConfigValue .....	302
7.2.1.85. Procedure: dbms.showCurrentUser .....	303
7.2.1.86. Procedure: dbms.upgrade .....	304
7.2.1.87. Procedure: dbms.upgradeStatus .....	305
7.2.1.88. Procedure: jwt.security.requestAccess .....	306
7.2.1.89. Procedure: tx.getMetaData .....	307
7.2.1.90. Procedure: tx.setMetaData .....	308
7.2.2. Functions .....	309
7.2.2.1. Function: abs .....	309
7.2.2.2. Function: abs .....	310
7.2.2.3. Function: acos .....	311
7.2.2.4. Function: all .....	312
7.2.2.5. Function: any .....	313
7.2.2.6. Function: asin .....	314
7.2.2.7. Function: atan .....	315
7.2.2.8. Function: atan2 .....	316
7.2.2.9. Function: avg .....	317
7.2.2.10. Function: avg .....	318
7.2.2.11. Function: avg .....	319
7.2.2.12. Function: ceil .....	320
7.2.2.13. Function: coalesce .....	321
7.2.2.14. Function: collect .....	322
7.2.2.15. Function: cos .....	323
7.2.2.16. Function: cot .....	324
7.2.2.17. Function: count .....	325
7.2.2.18. Function: date .....	326
7.2.2.19. Function: date.realtime .....	327
7.2.2.20. Function: date.statement .....	328
7.2.2.21. Function: date.transaction .....	329
7.2.2.22. Function: date.truncate .....	330
7.2.2.23. Function: datetime .....	331
7.2.2.24. Function: datetime.fromepoch .....	332
7.2.2.25. Function: datetime.fromepochmillis .....	333
7.2.2.26. Function: datetime.realtime .....	334
7.2.2.27. Function: datetime.statement .....	335
7.2.2.28. Function: datetime.transaction .....	336
7.2.2.29. Function: datetime.truncate .....	337
7.2.2.30. Function: degrees .....	338
7.2.2.31. Function: distance .....	339

7.2.2.32. Function: duration .....	340
7.2.2.33. Function: duration.between .....	341
7.2.2.34. Function: duration.inDays .....	342
7.2.2.35. Function: duration.inMonths .....	343
7.2.2.36. Function: duration.inSeconds .....	344
7.2.2.37. Function: e .....	345
7.2.2.38. Function: endNode .....	346
7.2.2.39. Function: exists .....	347
7.2.2.40. Function: exp .....	348
7.2.2.41. Function: file .....	349
7.2.2.42. Function: floor .....	350
7.2.2.43. Function: haversin .....	351
7.2.2.44. Function: head .....	352
7.2.2.45. Function: id .....	353
7.2.2.46. Function: id .....	354
7.2.2.47. Function: isEmpty .....	355
7.2.2.48. Function: isEmpty .....	356
7.2.2.49. Function: isEmpty .....	357
7.2.2.50. Function: keys .....	358
7.2.2.51. Function: keys .....	359
7.2.2.52. Function: keys .....	360
7.2.2.53. Function: labels .....	361
7.2.2.54. Function: last .....	362
7.2.2.55. Function: left .....	363
7.2.2.56. Function: length .....	364
7.2.2.57. Function: linenumber .....	365
7.2.2.58. Function: localdatetime .....	366
7.2.2.59. Function: localdatetime.realtime .....	367
7.2.2.60. Function: localdatetime.statement .....	368
7.2.2.61. Function: localdatetime.transaction .....	369
7.2.2.62. Function: localdatetime.truncate .....	370
7.2.2.63. Function: localtime .....	371
7.2.2.64. Function: localtime.realtime .....	372
7.2.2.65. Function: localtime.statement .....	373
7.2.2.66. Function: localtime.transaction .....	374
7.2.2.67. Function: localtime.truncate .....	375
7.2.2.68. Function: log .....	376
7.2.2.69. Function: log10 .....	377
7.2.2.70. Function: ltrim .....	378
7.2.2.71. Function: max .....	379
7.2.2.72. Function: min .....	380
7.2.2.73. Function: nodes .....	381
7.2.2.74. Function: none .....	382
7.2.2.75. Function: percentileCont .....	383
7.2.2.76. Function: percentileDisc .....	384
7.2.2.77. Function: percentileDisc .....	385

7.2.2.78. Function: pi .....	386
7.2.2.79. Function: point .....	387
7.2.2.80. Function: properties .....	388
7.2.2.81. Function: properties .....	389
7.2.2.82. Function: properties .....	390
7.2.2.83. Function: radians .....	391
7.2.2.84. Function: rand .....	392
7.2.2.85. Function: randomUUID .....	393
7.2.2.86. Function: range .....	394
7.2.2.87. Function: range .....	395
7.2.2.88. Function: reduce .....	396
7.2.2.89. Function: relationships .....	397
7.2.2.90. Function: replace .....	398
7.2.2.91. Function: reverse .....	399
7.2.2.92. Function: reverse .....	400
7.2.2.93. Function: right .....	401
7.2.2.94. Function: round .....	402
7.2.2.95. Function: round .....	403
7.2.2.96. Function: round .....	404
7.2.2.97. Function: rtrim .....	405
7.2.2.98. Function: sign .....	406
7.2.2.99. Function: sign .....	407
7.2.2.100. Function: sin .....	408
7.2.2.101. Function: single .....	409
7.2.2.102. Function: size .....	410
7.2.2.103. Function: size .....	411
7.2.2.104. Function: split .....	412
7.2.2.105. Function: split .....	413
7.2.2.106. Function: sqrt .....	414
7.2.2.107. Function: startNode .....	415
7.2.2.108. Function: stdev .....	416
7.2.2.109. Function: stdevp .....	417
7.2.2.110. Function: substring .....	418
7.2.2.111. Function: substring .....	419
7.2.2.112. Function: sum .....	420
7.2.2.113. Function: sum .....	421
7.2.2.114. Function: sum .....	422
7.2.2.115. Function: tail .....	423
7.2.2.116. Function: tan .....	424
7.2.2.117. Function: time .....	425
7.2.2.118. Function: time.realtime .....	426
7.2.2.119. Function: time.statement .....	427
7.2.2.120. Function: time.transaction .....	428
7.2.2.121. Function: time.truncate .....	429
7.2.2.122. Function: toBoolean .....	430
7.2.2.123. Function: toBoolean .....	431

7.2.2.124. Function: toBoolean .....	432
7.2.2.125. Function: toBooleanList .....	433
7.2.2.126. Function: toBooleanOrNull .....	434
7.2.2.127. Function: toFloat .....	435
7.2.2.128. Function: toFloat .....	436
7.2.2.129. Function: toFloatList .....	437
7.2.2.130. Function: toFloatOrNull .....	438
7.2.2.131. Function: toInteger .....	439
7.2.2.132. Function: toInteger .....	440
7.2.2.133. Function: toInteger .....	441
7.2.2.134. Function: toIntegerList .....	442
7.2.2.135. Function: toIntegerOrNull .....	443
7.2.2.136. Function: toLower .....	444
7.2.2.137. Function: toString .....	445
7.2.2.138. Function: toStringList .....	446
7.2.2.139. Function: toStringOrNull .....	447
7.2.2.140. Function: toUpper .....	448
7.2.2.141. Function: trim .....	449
7.2.2.142. Function: type .....	450
8. Sample SQL Database .....	452
8.1. Business Entities .....	453
8.2. People .....	455
8.2.1. Tables .....	456
8.2.1.1. Table: Person.Address .....	456
8.2.1.2. Table: Person.AddressType .....	458
8.2.1.3. Table: Person.BusinessEntity .....	459
8.2.1.4. Table: Person.BusinessEntityAddress .....	460
8.2.1.5. Table: Person.BusinessEntityContact .....	462
8.2.1.6. Table: Person.ContactType .....	464
8.2.1.7. Table: Person.CountryRegion .....	465
8.2.1.8. Table: Person.EmailAddress .....	466
8.2.1.9. Table: Person.Password .....	467
8.2.1.10. Table: Person.Person .....	468
8.2.1.11. Table: Person.PersonPhone .....	471
8.2.1.12. Table: Person.PhoneNumberType .....	472
8.2.1.13. Table: Person.StateProvince .....	473
8.2.2. Views .....	475
8.2.2.1. View: Person.vAdditionalContactInfo .....	475
8.2.2.2. View: Person.vStateProvinceCountryRegion .....	477
8.3. Human Resources .....	479
8.3.1. Tables .....	480
8.3.1.1. Table: HumanResources.Department .....	480
8.3.1.2. Table: HumanResources.Employee .....	481
8.3.1.3. Table: HumanResources.EmployeeDepartmentHistory .....	484
8.3.1.4. Table: HumanResources.EmployeePayHistory .....	486
8.3.1.5. Table: HumanResources.JobCandidate .....	487

8.3.1.6. Table: HumanResources.Shift .....	488
8.3.2. Views .....	489
8.3.2.1. View: HumanResources.vEmployee .....	489
8.3.2.2. View: HumanResources.vEmployeeDepartment .....	491
8.3.2.3. View: HumanResources.vEmployeeDepartmentHistory .....	492
8.3.2.4. View: HumanResources.vJobCandidate .....	493
8.3.2.5. View: HumanResources.vJobCandidateEducation .....	495
8.3.2.6. View: HumanResources.vJobCandidateEmployment .....	497
8.3.3. Procedures .....	499
8.3.3.1. Procedure: dbo.uspGetEmployeeManagers .....	499
8.3.3.2. Procedure: dbo.uspGetManagerEmployees .....	500
8.3.3.3. Procedure: dbo.uspSearchCandidateResumes .....	501
8.3.3.4. Procedure: HumanResources.uspUpdateEmployeeHireInfo .....	502
8.3.3.5. Procedure: HumanResources.uspUpdateEmployeeLogin .....	503
8.3.3.6. Procedure: HumanResources.uspUpdateEmployeePersonalInfo .....	504
8.3.4. Functions .....	505
8.3.4.1. Function: dbo.ufnGetContactInformation .....	505
8.4. Products .....	508
8.4.1. Tables .....	509
8.4.1.1. Table: Production.Culture .....	509
8.4.1.2. Table: Production.Document .....	510
8.4.1.3. Table: Production.Illustration .....	512
8.4.1.4. Table: Production.Product .....	513
8.4.1.5. Table: Production.ProductCategory .....	517
8.4.1.6. Table: Production.ProductDescription .....	518
8.4.1.7. Table: Production.ProductDocument .....	519
8.4.1.8. Table: Production.ProductModel .....	520
8.4.1.9. Table: Production.ProductModelIllustration .....	522
8.4.1.10. Table: Production.ProductModelProductDescriptionCulture .....	523
8.4.1.11. Table: Production.ProductPhoto .....	524
8.4.1.12. Table: Production.ProductProductPhoto .....	525
8.4.1.13. Table: Production.ProductReview .....	526
8.4.1.14. Table: Production.ProductSubcategory .....	527
8.4.1.15. Table: Production.UnitMeasure .....	529
8.4.2. Views .....	530
8.4.2.1. View: Production.vProductAndDescription .....	530
8.4.2.2. View: Production.vProductModelCatalogDescription .....	531
8.4.2.3. View: Production.vProductModelInstructions .....	533
8.4.3. Procedures .....	534
8.4.3.1. Procedure: dbo.uspGetBillOfMaterials .....	534
8.4.3.2. Procedure: dbo.uspGetWhereUsedProductID .....	535
8.4.4. Functions .....	536
8.4.4.1. Function: dbo.ufnGetDocumentStatusText .....	536
8.4.4.2. Function: dbo.ufnGetProductDealerPrice .....	537
8.4.4.3. Function: dbo.ufnGetProductListPrice .....	538
8.4.4.4. Function: dbo.ufnGetProductStandardCost .....	539

8.4.4.5. Function: dbo.ufnGetStock .....	540
8.5. Manufacturing .....	542
8.5.1. Tables .....	543
8.5.1.1. Table: Production.BillOfMaterials .....	543
8.5.1.2. Table: Production.ProductCostHistory .....	545
8.5.1.3. Table: Production.ProductListPriceHistory .....	546
8.5.1.4. Table: Production.ScrapReason .....	547
8.5.1.5. Table: Production.TransactionHistory .....	548
8.5.1.6. Table: Production.TransactionHistoryArchive .....	549
8.5.1.7. Table: Production.WorkOrder .....	550
8.5.1.8. Table: Production.WorkOrderRouting .....	553
8.6. Purchasing .....	555
8.6.1. Tables .....	556
8.6.1.1. Table: Purchasing.ProductVendor .....	556
8.6.1.2. Table: Purchasing.PurchaseOrderDetail .....	558
8.6.1.3. Table: Purchasing.PurchaseOrderHeader .....	562
8.6.1.4. Table: Purchasing.ShipMethod .....	565
8.6.1.5. Table: Purchasing.Vendor .....	566
8.6.2. Views .....	568
8.6.2.1. View: Purchasing.vVendorWithAddresses .....	568
8.6.2.2. View: Purchasing.vVendorWithContacts .....	569
8.6.3. Functions .....	571
8.6.3.1. Function: dbo.ufnGetPurchaseOrderStatusText .....	571
8.7. Inventory .....	573
8.7.1. Tables .....	574
8.7.1.1. Table: Production.Location .....	574
8.7.1.2. Table: Production.ProductInventory .....	575
8.8. Sales .....	577
8.8.1. Tables .....	578
8.8.1.1. Table: Sales.CountryRegionCurrency .....	578
8.8.1.2. Table: Sales.CreditCard .....	579
8.8.1.3. Table: Sales.Currency .....	580
8.8.1.4. Table: Sales.CurrencyRate .....	581
8.8.1.5. Table: Sales.Customer .....	583
8.8.1.6. Table: Sales.PersonCreditCard .....	585
8.8.1.7. Table: Sales.SalesOrderDetail .....	586
8.8.1.8. Table: Sales.SalesOrderHeader .....	589
8.8.1.9. Table: Sales.SalesOrderHeaderSalesReason .....	593
8.8.1.10. Table: Sales.SalesPerson .....	594
8.8.1.11. Table: Sales.SalesPersonQuotaHistory .....	596
8.8.1.12. Table: Sales.SalesReason .....	597
8.8.1.13. Table: Sales.SalesTaxRate .....	598
8.8.1.14. Table: Sales.SalesTerritory .....	599
8.8.1.15. Table: Sales.SalesTerritoryHistory .....	601
8.8.1.16. Table: Sales.ShoppingCartItem .....	602
8.8.1.17. Table: Sales.SpecialOffer .....	603

8.8.18. Table: Sales.SpecialOfferProduct .....	604
8.8.19. Table: Sales.Store .....	605
8.8.2. Views .....	607
8.8.2.1. View: Sales.vIndividualCustomer .....	607
8.8.2.2. View: Sales.vPersonDemographics .....	609
8.8.2.3. View: Sales.vSalesPerson .....	611
8.8.2.4. View: Sales.vSalesPersonSalesByFiscalYears .....	613
8.8.2.5. View: Sales.vStoreWithAddresses .....	614
8.8.2.6. View: Sales.vStoreWithContacts .....	615
8.8.2.7. View: Sales.vStoreWithDemographics .....	617
8.8.3. Functions .....	619
8.8.3.1. Function: dbo.ufnGetAccountingEndDate .....	619
8.8.3.2. Function: dbo.ufnGetAccountingStartDate .....	620
8.8.3.3. Function: dbo.ufnGetSalesOrderStatusText .....	621
8.8.3.4. Function: dbo.ufnLeadingZeros .....	622
8.9. Admin .....	624
8.9.1. Tables .....	624
8.9.1.1. Table: dbo.AWBuildVersion .....	624
8.9.1.2. Table: dbo.DatabaseLog .....	625
8.9.1.3. Table: dbo.ErrorLog .....	626
8.9.2. Procedures .....	627
8.9.2.1. Procedure: dbo.uspLogError .....	627
8.9.2.2. Procedure: dbo.uspPrintError .....	629
9. Sample SSAS Tabular Model .....	631
9.1. Internet Sales .....	632
9.1.1. Tables .....	633
9.1.1.1. Table: Customer .....	633
9.1.1.2. Table: Date .....	635
9.1.1.3. Table: Geography .....	637
9.1.1.4. Table: Internet Sales .....	638
9.1.1.5. Table: Product .....	640
9.1.1.6. Table: Product Category .....	642
9.1.1.7. Table: Product Subcategory .....	643

## Legend

- 🔑 Primary key
- 🔑 Primary key disabled
- 🔑 User-defined primary key
- 🔑 Unique key
- 🔑 Unique key disabled
- 🔑 User-defined unique key
- ⚡ Active trigger
- ⚡ Disabled trigger
- Many to one relation
- User-defined many to one relation
- ← One to many relation
- ↖ User-defined one to many relation
- ↔ Many to many relation
- ↙ User-defined many to many relation
- One to one relation
- User-defined one to one relation
- ↗ Input
- ↘ Output
- ↗ User/Output
- ↗ Uses dependency
- ↗ User-defined uses dependency
- ↘ Used by dependency
- ↙ User-defined used by dependency

TRIAL

## 1. Sample Business Glossary

This is a sample **Business Glossary** of fictitious organization.

And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this glossary in your production environment.

## 1.1. Business Glossary

### 1.1. Terms

#### 1.1.1. Term: Bill Of Materials

**Sample field: Status:** Active

Items required to make bicycles and bicycle subassemblies. It identifies the hierarchical relationship between a parent product and its components.

#### Relationships

Relationship	Related term
Is child of	 Manufacturing
Is contained in	 Product

#### Data Links

Object	Documentation
 Production.BillOfMaterials	 Sample SQL Database

## 1.1.2. Term: Billing Address

**Sample field: Status:** Active

Customer billing address provided on the sales order.

### Relationships

Relationship	Related term
Is child of	Sales Order
Is contained in	Sales Order

### Data Links

Object	Documentation
SalesOrderHeader.BillToAddressID	Sample SQL Database

TRIAL

### 1.1.3. Term: Company

#### Relationships

Relationship	Related term
Is parent of	Company name
Is parent of	Ticker symbol

#### Data Links

Object	Documentation
 2015Fortune500.csv (Fortune 500 Companies (2015))	 Sample Data Lake
 Fortune500.csv (Fortune 500 Companies (2016))	 Sample Data Lake

TRIAL

#### 1.1.4. Term: Company name

##### Relationships

Relationship	Related term
Is child of	 Company

##### Data Links

Object	Documentation
 2015Fortune500.csv.Company	 Sample Data Lake
 company_tickers.json.N.title (Company name)	 Sample Data Lake
 Fortune500.csv.COMPANY	 Sample Data Lake

TRIAL

## 1.1.5. Term: Customer

**Sample field: Status:** Active

Company customer.

### Relationships

Relationship	Related term
Related term	 Vendor

### Data Links

Object	Documentation
 Sales.Customer	 Sample SQL Database

TRIAL

## 1.1.6. Term: Customer Product Review

**Sample field: Status:** Active

Customer review of the product they have purchased. It includes a rating (1-5) and comments.

### Relationships

Relationship	Related term
Is child of	 Product

### Data Links

Object	Documentation
 Production.ProductReview	 Sample SQL Database

TRIAL

## 1.1.7. Term: Customer Purchase Order Number

**Sample field: Status:** Active

**Also known as:** Customer PO No

Customer purchase order number reference on sales order.

### Relationships

Relationship	Related term
Is child of	 Sales Order
Is contained in	 Sales Order

### Data Links

Object	Documentation
 SalesOrderHeader.PurchaseOrderNumber	 Sample SQL Database

TRIAL

## 1.1.8. Term: Department

**Sample field: Status:** Active

Departments within the organization.

### Data Links

Object	Documentation
 HumanResources.Department	 Sample SQL Database

TRIAL

## 1.1.9. Term: Freight

**Sample field: Status:** Active

PO shipping cost.

### Data Links

Object	Documentation
PurchaseOrderHeader.Freight	 Sample SQL Database
SalesOrderHeader.Freight	 Sample SQL Database

TRIAL

## 1.1.10. Term: Inventory Location

**Sample field: Status:** Active

Location in a warehouse - example: shelf, bin.

### Relationships

Relationship	Related term
Is child of	 Inventory
Related term	 Manufacturing Location
Is contained in	 Warehouse

### Data Links

Object	Documentation
 Production.ProductInventory	 Sample SQL Database
 ProductInventory.LocationID	 Sample SQL Database

TRIAL

## 1.1.11. Term: Manufacturing Location

**Sample field: Status:** Active

Product manufacturing locations.

### Relationships

Relationship	Related term
Related term	Inventory Location
Is child of	Manufacturing
Contains	Mfg Location Capacity
Is parent of	Mfg Location Capacity
Contains	Mfg Location Cost Rate
Is parent of	Mfg Location Cost Rate
Related term	Warehouse

### Data Links

Object	Documentation
Production.Location	Sample SQL Database

## 1.1.12. Term: Mfg Location Capacity

**Sample field: Status:** Active

The standard hourly cost of the manufacturing location.

### Relationships

Relationship	Related term
Is child of	 Manufacturing Location
Is contained in	 Manufacturing Location

### Data Links

Object	Documentation
 Location.Availability	 Sample SQL Database

TRIAL

### 1.1.13. Term: Mfg Location Cost Rate

**Sample field: Status:** Active

Work capacity (in hours) of the manufacturing location.

#### Relationships

Relationship	Related term
Is child of	 Manufacturing Location
Is contained in	 Manufacturing Location

#### Data Links

Object	Documentation
 Location.CostRate	 Sample SQL Database

TRIAL

## 1.1.14. Term: Online (Sales) Order

**Sample field: Status:** Active

Order placed online by the customer.

### Relationships

Relationship	Related term
Is a type of	 Sales Order
Is child of	 Sales Order
Contains	 Sales Order Number

### Data Links

Object	Documentation
 SalesOrderHeader.OnlineOrderFlag	 Sample SQL Database

TRIAL

## 1.1.15. Term: Product

**Sample field: Status:** Active

Products sold or used in the manufacturing of sold products.

### Relationships

Relationship	Related term
Contains	Bill Of Materials
Is parent of	Customer Product Review
Is contained in	Product Category
Is parent of	Product Category
Contains	Product Model
Is parent of	Product Model
Contains	Product Number
Is parent of	Product Number
Contains	Product Standard Cost
Is parent of	Product Standard Cost
Is contained in	Shopping Cart

### Data Links

Object	Documentation
Production.Product	Sample SQL Database

## 1.1.16. Term: Product Category

**Sample field: Status:** Active

High-level product categorization.

### Relationships

Relationship	Related term
Contains	 Product
Is child of	 Product

### Data Links

Object	Documentation
 Production.ProductCategory	 Sample SQL Database

TRIAL

### 1.1.17. Term: Product Model

**Sample field: Status:** Active

Product model classification.

#### Relationships

Relationship	Related term
Is child of	 Product
Is contained in	 Product

#### Data Links

Object	Documentation
 Production.ProductModel	 Sample SQL Database

TRIAL

## 1.1.18. Term: Product Number

**Sample field: Status:** Active

Unique product identification number.

### Relationships

Relationship	Related term
Is child of	 Product
Is contained in	 Product

### Data Links

Object	Documentation
 Product.ProductNumber	 Sample SQL Database

TRIAL

## 1.1.19. Term: Product Standard Cost

**Sample field: Status:** Active

The standard cost of the product.

### Relationships

Relationship	Related term
Is child of	 Product
Is contained in	 Product

### Data Links

Object	Documentation
 Product.StandardCost	 Sample SQL Database
 Production.ProductCostHistory	 Sample SQL Database
 ProductCostHistory.StandardCost	 Sample SQL Database

TRIAL

## 1.1.20. Term: Purchase Order

**Sample field: Status:** Active

Purchase order placed by the organization to the vendor as part of the organization procurement department activity.

### Relationships

Relationship	Related term
Is child of	Procurement
Contains	Purchase Order Date
Is parent of	Purchase Order Date
Contains	Purchase Order Due Date
Is parent of	Purchase Order Due Date
Contains	Purchase Order Quantity
Is parent of	Purchase Order Quantity
Contains	Purchase Order Received Quantity
Is parent of	Purchase Order Received Quantity
Contains	Purchase Order Rejected Quantity
Is parent of	Purchase Order Rejected Quantity
Contains	Purchase Order Ship Date
Is parent of	Purchase Order Ship Date
Contains	Purchase Order Stocked Quantity
Is parent of	Purchase Order Stocked Quantity
Related term	Sales Order

### Data Links

Object	Documentation
Purchasing.PurchaseOrderDetail	Sample SQL Database
Purchasing.PurchaseOrderHeader	Sample SQL Database

## 1.1.21. Term: Purchase Order Date

**Sample field: Status:** Active

The date purchase order was created in the purchasing management system.

### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order
Related term	 Sales Order Date

### Data Links

Object	Documentation
 PurchaseOrderHeader.OrderDate	 Sample SQL Database

TRIAL

## 1.1.22. Term: Purchase Order Due Date

**Sample field: Status:** Active

The date the purchased product is expected to be received from the vendor.

### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order
Related term	 Sales Order Due Date

### Data Links

Object	Documentation
 PurchaseOrderDetail.DueDate	 Sample SQL Database

TRIAL

### 1.1.23. Term: Purchase Order Quantity

Quantity ordered on purchase order.

#### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order

#### Data Links

Object	Documentation
 PurchaseOrderDetail.OrderQty	 Sample SQL Database

TRIAL

## 1.1.24. Term: Purchase Order Received Quantity

**Sample field: Status:** Active

Purchase order item quantity actually received from the vendor.

### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order
Related term	 Purchase Order Rejected Quantity
Is used to calculate	 Purchase Order Stocked Quantity

### Data Links

Object	Documentation
 PurchaseOrderDetail.ReceivedQty	 Sample SQL Database

TRIAL

## 1.1.25. Term: Purchase Order Rejected Quantity

**Sample field: Status:** Active

Purchase order item quantity rejected during inspection to the vendor.

### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order
Related term	 Purchase Order Received Quantity
Is used to calculate	 Purchase Order Stocked Quantity

### Data Links

Object	Documentation
 PurchaseOrderDetail.RejectedQty	 Sample SQL Database

TRIAL

## 1.1.26. Term: Purchase Order Ship Date

**Sample field: Status:** Active

Estimated shipment date from the vendor.

### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order
Related term	 Sales Order Ship Date

### Data Links

Object	Documentation
 PurchaseOrderHeader.ShipDate	 Sample SQL Database

TRIAL

## 1.1.27. Term: Purchase Order Stocked Quantity

**Sample field: Status:** Active

Purchase order item quantity accepted into inventory and computed as Received Quantity - Rejected Quantity.

### Relationships

Relationship	Related term
Is child of	 Purchase Order
Is contained in	 Purchase Order
Is calculated from	 Purchase Order Received Quantity
Is calculated from	 Purchase Order Rejected Quantity

### Data Links

Object	Documentation
 PurchaseOrderDetail.StockedQty	 Sample SQL Database

TRIAL

## 1.1.28. Term: Sales Bonus

**Sample field: Status:** Active

A specific dollar amount of bonus due for specific salesperson if the quota is met.

### Relationships

Relationship	Related term
Is calculated from	 Sales Commission
Is child of	 Sales Person
Is calculated from	 Sales Person Quota
Is calculated from	 Sales Person Sales

### Data Links

Object	Documentation
 SalesPerson.Bonus	 Sample SQL Database

TRIAL

## 1.1.29. Term: Sales Commission

**Sample field: Status:** Active

Commission percent received per sale by specific salesperson.

### Relationships

Relationship	Related term
Is used to calculate	 Sales Bonus
Is child of	 Sales Person

### Data Links

Object	Documentation
 SalesPerson.CommissionPct	 Sample SQL Database

TRIAL

## 1.1.30. Term: Sales Order

**Sample field: Status:** Active

Sales order placed by customer.

### Relationships

Relationship	Related term
Contains	Billing Address
Is parent of	Billing Address
Contains	Customer Purchase Order Number
Is parent of	Customer Purchase Order Number
Has a type	Online (Sales) Order
Is parent of	Online (Sales) Order
Related term	Purchase Order
Is child of	Sales
Contains	Sales Order Date
Is parent of	Sales Order Date
Contains	Sales Order Due Date
Is parent of	Sales Order Due Date
Contains	Sales Order Number
Is parent of	Sales Order Number
Contains	Sales Order Ship Date
Is parent of	Sales Order Ship Date
Is parent of	Sales Reason
Is parent of	Shipping Address

### Data Links

Object	Documentation
Sales.SalesOrderDetail	Sample SQL Database
Sales.SalesOrderHeader	Sample SQL Database

### 1.1.31. Term: Sales Order Date

**Sample field: Status:** Active

A date the sales order was created.

#### Relationships

Relationship	Related term
Related term	Purchase Order Date
Is child of	Sales Order
Is contained in	Sales Order
Related term	Sales Order Due Date
Related term	Sales Order Ship Date

#### Data Links

Object	Documentation
SalesOrderHeader.OrderDate	Sample SQL Database

## 1.1.32. Term: Sales Order Due Date

**Sample field: Status:** Active

A date sales order is due to the customer.

### Relationships

Relationship	Related term
Related term	Purchase Order Due Date
Is child of	Sales Order
Is contained in	Sales Order
Related term	Sales Order Date
Related term	Sales Order Ship Date

### Data Links

Object	Documentation
SalesOrderHeader.DueDate	Sample SQL Database

### 1.1.33. Term: Sales Order Number

**Sample field: Status:** Active

Unique sales order identification number in format 'SO<integer no>', for instance 'SO46146'.

#### Relationships

Relationship	Related term
Is contained in	 Online (Sales) Order
Is child of	 Sales Order
Is contained in	 Sales Order

#### Data Links

Object	Documentation
 SalesOrderHeader.SalesOrderNumber	 Sample SQL Database

TRIAL

## 1.1.34. Term: Sales Order Ship Date

**Sample field: Status:** Active

A date sales order was shipped to the customer.

### Relationships

Relationship	Related term
Related term	Purchase Order Ship Date
Is child of	Sales Order
Is contained in	Sales Order
Related term	Sales Order Date
Related term	Sales Order Due Date

### Data Links

Object	Documentation
SalesOrderHeader.ShipDate	Sample SQL Database

TRIAL

### 1.1.35. Term: Sales Person

**Sample field: Status:** Active

Organization sales representative.

#### Relationships

Relationship	Related term
Is child of	Sales
Is parent of	Sales Bonus
Is parent of	Sales Commission
Is parent of	Sales Person Quota
Is parent of	Sales Person Sales

#### Data Links

Object	Documentation
Sales.SalesPerson	Sample SQL Database

TRIAL

## 1.1.36. Term: Sales Person Quota

**Sample field: Status:** Active

Projected yearly sales for a specific salesperson.

### Relationships

Relationship	Related term
Is used to calculate	 Sales Bonus
Is child of	 Sales Person

### Data Links

Object	Documentation
 Sales.SalesPersonQuotaHistory	 Sample SQL Database

TRIAL

## 1.1.37. Term: Sales Person Sales

**Sample field: Status:** Active

Value of total value of sales of a salesperson in a specific time period.

### Relationships

Relationship	Related term
Is used to calculate	 Sales Bonus
Is child of	 Sales Person

### Data Links

Object	Documentation
 SalesPerson.SalesLastYear	 Sample SQL Database
 SalesPerson.SalesYTD	 Sample SQL Database

TRIAL

### 1.1.38. Term: Sales Reason

**Sample field: Status:** Active

Customer purchase reason.

#### Relationships

Relationship	Related term
Is child of	Sales Order

#### Data Links

Object	Documentation
Sales.SalesReason	Sample SQL Database

TRIAL

### 1.1.39. Term: Sales Territory

**Sample field: Status:** Active

The geographical area that groups customers and salespeople.

#### Relationships

Relationship	Related term
Is child of	Sales
Contains	Store

#### Data Links

Object	Documentation
Sales.SalesTerritory	Sample SQL Database

TRIAL

### 1.1.40. Term: Shift

**Sample field: Status:** Active

Manufacturing work shift.

#### Relationships

Relationship	Related term
Is child of	 Manufacturing

#### Data Links

Object	Documentation
 HumanResources.Shift	 Sample SQL Database

TRIAL

### 1.1.41. Term: Ship Base

**Sample field: Status:** Active

Shipping company minimum shipping charge.

#### Relationships

Relationship	Related term
Is child of	 Procurement

#### Data Links

Object	Documentation
 ShipMethod.ShipBase	 Sample SQL Database

TRIAL

### 1.1.42. Term: Ship Method

**Sample field: Status:** Active

Shipping company used for purchasing and sales orders shipments.

#### Relationships

Relationship	Related term
Is child of	 Procurement

#### Data Links

Object	Documentation
 Purchasing.ShipMethod	 Sample SQL Database

TRIAL

### 1.1.43. Term: Ship Rate

**Sample field: Status:** Active

Shipping company shipping charge per pound.

#### Relationships

Relationship	Related term
Is child of	 Procurement

#### Data Links

Object	Documentation
 ShipMethod.ShipRate	 Sample SQL Database

TRIAL

## 1.1.44. Term: Shipping Address

**Sample field: Status:** Active

Customer shipping address provided on the sales order.

### Relationships

Relationship	Related term
Is child of	 Sales Order

### Data Links

Object	Documentation
 SalesOrderHeader.ShipToAddressID	 Sample SQL Database

TRIAL

## 1.1.45. Term: Shopping Cart

**Sample field: Status:** Active

A virtual cart that contains online customer orders until the order is submitted or canceled.

### Relationships

Relationship	Related term
Contains	 Product
Is child of	 Sales

### Data Links

Object	Documentation
 Sales.ShoppingCartItem	 Sample SQL Database

TRIAL

### 1.1.46. Term: Store

**Sample field: Status:** Active

Organization reseller.

#### Relationships

Relationship	Related term
Is child of	Sales
Is contained in	Sales Territory

#### Data Links

Object	Documentation
Sales.Store	Sample SQL Database

TRIAL

### 1.1.47. Term: Ticker symbol

A **ticker symbol** or **stock symbol** is an abbreviation used to uniquely identify publicly traded shares of a particular stock on a particular stock market.

#### Relationships

Relationship	Related term
Is child of	Company

#### Data Links

Object	Documentation
company_tickers.json (Company tickers)	Sample Data Lake
company_tickers.json.N.ticker (Company ticker)	Sample Data Lake
Fortune500.csv.SYMBOL	Sample Data Lake

TRIAL

## 1.1.48. Term: Vendor

**Sample field: Status:** Active

Companies from whom the organization purchases parts or other goods.

### Relationships

Relationship	Related term
Related term	 Customer
Is child of	 Procurement
Contains	 Vendor Credit Rating
Is parent of	 Vendor Credit Rating

### Data Links

Object	Documentation
 Purchasing.ProductVendor	 Sample SQL Database
 Purchasing.Vendor	 Sample SQL Database

TRIAL

### 1.1.49. Term: Vendor Credit Rating

**Sample field: Status:** Active

Vendor credit rating. Closed list: Superior, Excellent, Above average, Average, Below average.

#### Relationships

Relationship	Related term
Is child of	 Vendor
Is contained in	 Vendor

#### Data Links

Object	Documentation
 Vendor.CreditRating	 Sample SQL Database

TRIAL

## 1.1.50. Term: Warehouse

**Sample field: Status:** Active

Product inventory location.

### Relationships

Relationship	Related term
Is child of	 Inventory
Contains	 Inventory Location
Related term	 Manufacturing Location

### Data Links

Object	Documentation
 Production.Location	 Sample SQL Database

TRIAL

## 1.1.51. Term: Work Order

**Sample field: Status:** Active

Manufacturing work orders.

### Relationships

Relationship	Related term
Is child of	 Manufacturing

### Data Links

Object	Documentation
 Production.WorkOrder	 Sample SQL Database

TRIAL

## 1.2. Categories

### 1.2.1. Category: Inventory

**Sample field: Status:** Active

Terms related to inventory management.

#### Relationships

Relationship	Related term
Is parent of	 Inventory Location
Is parent of	 Warehouse

TRIAL

## 1.2.2. Category: Manufacturing

**Sample field: Status:** Active

Terms related to manufacturing.

### Relationships

Relationship	Related term
Is parent of	 Bill Of Materials
Is parent of	 Manufacturing Location
Is parent of	 Shift
Is parent of	 Work Order

TRIAL

### 1.2.3. Category: Procurement

**Sample field: Status:** Active

Terms related to procurement.

#### Relationships

Relationship	Related term
Is parent of	 Purchase Order
Is parent of	 Ship Base
Is parent of	 Ship Method
Is parent of	 Ship Rate
Is parent of	 Vendor

TRIAL

#### 1.2.4. Category: Sales

**Sample field: Status:** Active

Terms related to sales.

##### Relationships

Relationship	Related term
Is parent of	Sales Order
Is parent of	Sales Person
Is parent of	Sales Territory
Is parent of	Shopping Cart
Is parent of	Store

TRIAL

## 2. ExamSystem

TRIAL

## 2.1. Data Dictionary

### 2.1.1. Tables

#### 2.1.1.1. Table: Choice

##### Columns

		Name	Data type	Description / Attributes
		Cho_Id	int	Identity / Auto increment
		Ques_Id	int	Nullable References: Question
		Cho_Content	varchar(100)	Nullable
		Cho_Char	varchar(1)	Nullable

##### Links to

Table		Join	Title / Name / Description
	Question	ChoiceQues_Id = QuestionQues_Id	FK_Choice_Question

##### Linked from

Table		Join	Title / Name / Description
	Question	ChoiceCho_Id = QuestionModel_Ans	FK_Question_Choice

##### Unique keys

Columns		Name / Description
	Cho_Id	PK_Choice

##### Uses

		Name
	Choice	
	Question	

##### Used By

		Name
	Choice	
	sp_CorrectStdExam	
	sp_deleteChoice	
	sp_ExamQuestionsAndAnswers	
	sp_InsertChoice	
	sp_selectchoice	
	sp_StdExamAnswers	
	sp_updateChoice	
	Question	

## 2.1.1.2. Table: Course

### Columns

	Name	Data type	Description / Attributes
█	Crs_Id	int	Identity / Auto increment
█	Crs_Name	varchar(50)	Nullable
█	Crs_Desc	varchar(50)	Nullable
█	Crs_Dur	int	Nullable

### Linked from

	Table	Join	Title / Name / Description
→	Crs_Top	CourseCrs_Id = Crs_TopCrs_Id	FK_Crs_Top_Course
→	Exam	CourseCrs_Id = ExamCrs_Id	FK_Exam_Course
→	Ins_Crs	CourseCrs_Id = Ins_CrsCrs_Id	FK_Ins_Crs_Course
→	Std_Crs	CourseCrs_Id = Std_CrsCrs_Id	FK_Std_Crs_Course

### Unique keys

	Columns	Name / Description
█	Crs_Id	PK_Course

### Used By

	Name
█	Course
⚙	sp_CourseNameWithStudentCount
⚙	sp_deleteCourse
⚙	sp_insertCourse
⚙	sp_selectCourse
⚙	sp_StudentGradePerCourse
⚙	sp_updateCourse
→	Crs_Top
→	Exam
→	Ins_Crs
→	Std_Crs

### 2.1.1.3. Table: Crs\_Top

#### Columns

	Name	Data type	Description / Attributes
█	Crs_Id	int	References: Course
█	Crs_Top	varchar(50)	

#### Links to

	Table	Join	Title / Name / Description
→	Course	Crs_TopCrs_Id = CourseCrs_Id	FK_Crs_Top_Course

#### Unique keys

	Columns	Name / Description
█	Crs_Id, Crs_Top	PK_Crs_Top

#### Uses

	Name
█	Crs_Top
→	Course

#### Used By

	Name
█	Crs_Top
⚙	sp_deleteTopic
⚙	sp_insertTopic
⚙	sp_selectTopic
⚙	sp_TopicsPerCourse
⚙	sp_updateTopic

## 2.1.1.4. Table: Department

### Columns

		Name	Data type	Description / Attributes
		Dept_Id	int	Identity / Auto increment
		Dept_Name	varchar(50)	Nullable
		Dept_Loc	varchar(50)	Nullable
		Dept_ManagerHireDate	date	Nullable
		Dept_ManagerId	int	Nullable References: Instructor

### Links to

Table		Join	Title / Name / Description
	Instructor	<b>Department</b> Dept_ManagerId = InstructorIns_Id	FK_Department_Instructor

### Linked from

Table		Join	Title / Name / Description
	Instructor	<b>Department</b> Dept_Id = InstructorDept_Id	FK_Instructor_Department
	Student	<b>Department</b> Dept_Id = StudentDept_Id	FK_Student_Department

### Unique keys

Columns		Name / Description
	Dept_Id	PK_Department

### Uses

Name	
	Department
	Instructor

### Used By

Name	
	Department
	sp_deleteDepartment
	sp_insertDepartment
	sp_selectDepartment
	sp_updateDepartment
	Instructor
	Student

## 2.1.1.5. Table: Exam

### Columns

		Name	Data type	Description / Attributes
		Exm_Id	int	Identity / Auto increment
		Crs_Id	int	Nullable References: Course
		Generator_Id	int	Nullable References: Instructor
		Exm_Grade	int	Nullable

### Links to

Table		Join	Title / Name / Description
	Course	<b>ExamCrs_Id = CourseCrs_Id</b>	FK_Exam_Course
	Instructor	<b>ExamGenerator_Id = InstructorIns_Id</b>	FK_Exam_Instructor

### Linked from

Table		Join	Title / Name / Description
	Exm_Ques	<b>ExamExm_Id = Exm_QuesExm_Id</b>	FK_Exm_Ques_Exam
	Std_Ques_Exm	<b>ExamExm_Id = Std_Ques_ExmExm_Id</b>	FK_Std_Ques_Exm_Exam

### Unique keys

Columns		Name / Description
	Exm_Id	PK_Exam

### Uses

		Name
	Exam	
	Course	
	Instructor	

### Used By

		Name
	Exam	
	sp_CorrectStdExam	
	sp_deleteExam	
	sp_GenerateExam	
	sp_insertExam	
	sp_selectExam	
	sp_updateExam	
	Exm_Ques	
	Std_Ques_Exm	

## 2.1.1.6. Table: Exm\_Ques

### Columns

		Name	Data type	Description / Attributes
		Exm_Id	int	<b>References:</b> Exam
		Ques_Id	int	<b>References:</b> Question
		Grade	int	<b>Nullable</b> <b>Default:</b> 1

### Links to

	Table	Join	Title / Name / Description
→	Exam	<b>Exm_Ques</b> Exm_Id = Exam.Exm_Id	FK_Exm_Ques_Exam
→	Question	<b>Exm_Ques</b> Ques_Id = Question.Ques_Id	FK_Exm_Ques_Question

### Unique keys

	Columns	Name / Description
	Exm_Id, Ques_Id	PK_Exm_Ques

### Uses

	Name
	<b>Exm_Ques</b>
→	Exam
→	Question

### Used By

	Name
	<b>Exm_Ques</b>
	sp_ExamQuestionsAndAnswers
	sp_GenerateExam

## 2.1.1.7. Table: Ins\_Crs

### Columns

	Name	Data type	Description / Attributes
	Ins_Id	int	<b>References:</b> Instructor
	Crs_Id	int	<b>References:</b> Course

### Links to

	Table	Join	Title / Name / Description
	Course	<b>Ins_Crs</b> Crs_Id = CourseCrs_Id	FK_Ins_Crs_Course
	Instructor	<b>Ins_Crs</b> Ins_Id = InstructorIns_Id	FK_Ins_Crs_Instructor

### Unique keys

	Columns	Name / Description
	Ins_Id, Crs_Id	PK_Ins_Crs

### Uses

	Name
	<b>Ins_Crs</b>
	Course
	Instructor

### Used By

	Name
	<b>Ins_Crs</b>
	DelCrsIns
	InsertInsForCrs
	SelectCrsIns
	sp_CourseNameWithStudentCount
	sp_UpdateCrsForIns

## 2.1.1.8. Table: Instructor

### Columns

	Name	Data type	Description / Attributes
█	Ins_Id	int	Identity / Auto increment
█	U_Id	int	Nullable References: User
█	Dept_Id	int	Nullable References: Department
█	Ins_Fname	varchar(50)	Nullable
█	Ins_Lname	varchar(50)	Nullable
█	Ins_Degree	varchar(50)	Nullable
█	Ins_Salary	int	Nullable

### Links to

	Table	Join	Title / Name / Description
→	Department	InstructorDept_Id = DepartmentDept_Id	FK_Instructor_Department
→	User	InstructorU_Id = UserU_Id	FK_Instructor_User

### Linked from

	Table	Join	Title / Name / Description
←	Department	InstructorIns_Id = DepartmentDept_ManagerId	FK_Department_Instructor
←	Exam	InstructorIns_Id = ExamGenerator_Id	FK_Exam_Instructor
←	Ins_Crs	InstructorIns_Id = Ins_CrsIns_Id	FK_Ins_Crs_Instructor

### Unique keys

	Columns	Name / Description
█	Ins_Id	PK_Instructor

### Uses

	Name
█	Instructor
→	Department
→	User

### Used By

	Name
█	Instructor
⚙️	DeleteInstructor
⚙️	InsertInstructor
⚙️	SelectInstructor
⚙️	UpdateInstructor

Name

→ Department

→ Exam

→ Ins\_Crs

TRIAL

## 2.1.1.9. Table: Question

### Columns

	Name	Data type	Description / Attributes
█	Ques_Id	int	Identity / Auto increment
█	Crs_Id	int	Nullable
█	Ques_Content	varchar(300)	Nullable
█	Ques_Grade	int	Nullable
█	Ques_Type	varchar(3)	Nullable
█	Model_Ans	int	Nullable References: Choice

### Links to

	Table	Join	Title / Name / Description
→	Choice	QuestionModel_Ans = ChoiceCho_Id	FK_Question_Choice

### Linked from

	Table	Join	Title / Name / Description
←	Choice	QuestionQues_Id = ChoiceQues_Id	FK_Choice_Question
←	Exm_Ques	QuestionQues_Id = Exm_QuesQues_Id	FK_Exm_Ques_Question
←	Std_Ques_Exm	QuestionQues_Id = Std_Ques_ExmQues_Id	FK_Std_Ques_Exm_Question

### Unique keys

	Columns	Name / Description
█	Ques_Id	PK_Question

### Uses

	Name
█	Question
→	Choice

### Used By

	Name
█	Question
⚙	InsertQues
⚙	sp_CorrectStdExam
⚙	sp_DeleteQues
⚙	sp_ExamQuestionsAndAnswers
⚙	sp_GenerateExam
⚙	sp_SelectQues
⚙	sp_StdExamAnswers
⚙	UpdateQues

Name

→ Choice

→ Exm\_Ques

→ Std\_Ques\_Exm

TRIAL

## 2.1.1.10. Table: Std\_Crs

### Columns

		Name	Data type	Description / Attributes
		Std_Id	int	<b>References:</b> Student
		Crs_Id	int	<b>References:</b> Course
		Grade	int	<b>Nullable</b>
		Date	date	<b>Nullable</b>

### Links to

Table		Join	Title / Name / Description
	Course	<b>Std_Crs</b> Crs_Id = CourseCrs_Id	FK_Std_Crs_Course
	Student	<b>Std_Crs</b> Std_Id = StudentStd_Id	FK_Std_Crs_Student

### Unique keys

Columns		Name / Description
	Std_Id, Crs_Id	PK_Std_Crs

### Uses

Name	
	<b>Std_Crs</b>
	Course
	Student

### Used By

Name	
	<b>Std_Crs</b>
	sp_CorrectStdExam
	sp_CourseNameWithStudentCount
	sp_DeleteStdFromCrs
	sp_InsertStdForCrs
	sp_SelectStdOfCrs
	sp_StudentGradePerCourse
	sp_UpdateCrsOfStd

## 2.1.1.11. Table: Std\_Ques\_Exm

### Columns

		Name	Data type	Description / Attributes
█	🔑	Std_Id	int	References: Student
█	🔑	Exm_Id	int	References: Exam
█	🔑	Ques_Id	int	References: Question
█		Std_Answer	varchar(1)	

### Links to

	Table	Join	Title / Name / Description
→	Exam	Std_Ques_ExmExm_Id = ExamExam_Id	FK_Std_Ques_Exm_Exam
→	Question	Std_Ques_ExmQues_Id = QuestionQues_Id	FK_Std_Ques_Exm_Question
→	Student	Std_Ques_ExmStd_Id = StudentStd_Id	FK_Std_Ques_Exm_Student

### Unique keys

	Columns	Name / Description
🔑	Std_Id, Exm_Id, Ques_Id	PK_Std_Ques_Exm

### Uses

	Name
█	Std_Ques_Exm
→	Exam
→	Question
→	Student

### Used By

	Name
█	Std_Ques_Exm
⚙️	sp_AssignAnsToStd
⚙️	sp_CorrectStdExam
⚙️	sp_InsertStdAnsForExm
⚙️	sp_SelectStdAnsForExm
⚙️	sp_StdExamAnswers

## 2.1.1.12. Table: Student

### Columns

	Name	Data type	Description / Attributes
█	Std_Id	int	Identity / Auto increment
█	U_Id	int	Nullable References: User
█	Dept_Id	int	Nullable References: Department
█	Std_Fname	varchar(50)	Nullable
█	Std_Lname	varchar(50)	Nullable
█	Std_BOD	date	Nullable
█	Std_Address	varchar(100)	Nullable

### Links to

	Table	Join	Title / Name / Description
→	Department	StudentDept_Id = DepartmentDept_Id	FK_Student_Department
→	User	StudentU_Id = UserU_Id	FK_Student_User

### Linked from

	Table	Join	Title / Name / Description
←	Std_Crs	StudentStd_Id = Std_CrsStd_Id	FK_Std_Crs_Student
←	Std_Ques_Exm	StudentStd_Id = Std_Ques_ExmStd_Id	FK_Std_Ques_Exm_Student

### Unique keys

	Columns	Name / Description
█	Std_Id	PK_Student

### Uses

	Name
█	Student
→	Department
→	User

### Used By

	Name
█	Student
⚙️	DeleteStdByld
⚙️	InsertNewStd
⚙️	SelectStdByld
⚙️	sp_StudentGradePerCourse
⚙️	sp_StudentperDepartment
⚙️	UpdateStdInfo

Name

→ Std\_Crs

→ Std\_Ques\_Exm

TRIAL

## 2.1.1.13. Table: User

### Columns

	Name	Data type	Description / Attributes
█	U_Id	int	Identity / Auto increment
█	U_UserName	varchar(50)	
█	U_Email	varchar(50)	Nullable
█	U_Password	varchar(50)	
█	U_Sex	varchar(1)	Nullable
█	U_IsStd	bit	

### Linked from

	Table	Join	Title / Name / Description
→	Instructor	UserU_Id = InstructorU_Id	FK_Instructor_User
→	Student	UserU_Id = StudentU_Id	FK_Student_User

### Unique keys

	Columns	Name / Description
█	U_Id	PK_User
█	U_Id	IX_User

### Used By

	Name
█	User
⚙️	DeleteUserByUserName
⚙️	InsertNewUser
⚙️	SelectUserById
⚙️	UpdateUser
→	Instructor
→	Student

## 2.1.2. Procedures

### 2.1.2.1. Procedure: DelCrsIns

#### Input/Output

	Name	Data type	Description
→@	ins_id	int	
→@	crs_id	int	

#### Uses

	Name
⚙️	DelCrsIns
📄	Ins_Crs

#### Script

```
CREATE PROC DelCrsIns
AS
    BEGIN TRY
        DELETE FROM Ins_Crs
        WHERE Ins_Id = @ins_id
        AND Crs_Id = @crs_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

## 2.1.2.2. Procedure: DeleteInstructor

### Input/Output

Name	Data type	Description
@ins_id	int	

### Uses

Name
>DeleteInstructor
Instructor

### Script

```
CREATE PROC DeleteInstructor @ins_id int
AS
    BEGIN TRY
        DELETE FROM Instructor
        WHERE Ins_Id = @ins_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

TRIAL

### 2.1.2.3. Procedure: DeleteStdById

#### Input/Output

Name	Data type	Description
@std_id	int	

#### Uses

Name
>DeleteStdById
Student

#### Script

```
CREATE PROC DeleteStdById
AS
BEGIN TRY
    DELETE FROM Student
    WHERE Std_Id = @std_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

TRIAL

#### 2.1.2.4. Procedure: DeleteUserByUserName

##### Input/Output

Name	Data type	Description
@userN	varchar(50)	

##### Uses

Name
>DeleteUserByUserName
User

##### Script

```
CREATE PROC DeleteUserByUserName @userName varchar(50)
AS
    BEGIN TRY
        DELETE FROM [User]
        WHERE U_UserName = @userName
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

TRIAL

## 2.1.2.5. Procedure: InsertInsForCrs

### Input/Output

	Name	Data type	Description
→@	ins_id	int	
→@	crs_id	int	

### Uses

	Name
⚙️	InsertInsForCrs
↳	Ins_Crs

### Script

```
CREATE PROC InsertInsForCrs
AS
    BEGIN TRY
        INSERT INTO Ins_Crs
        VALUES (@ins_id, @crs_id)
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

## 2.1.2.6. Procedure: InsertInstructor

### Input/Output

	Name	Data type	Description
→@	u_id	int	
→@	dept_id	int	
→@	fname	varchar(50)	
→@	lname	varchar(50)	
→@	deg	varchar(50)	
→@	sal	int	

### Uses

	Name
⚙️	InsertInstructor
grid	Instructor

### Script

```

CREATE PROC InsertInstructor
AS
BEGIN TRY
    INSERT INTO Instructor
    VALUES (
        @u_id,
        @dept_id,
        @fname,
        @lname,
        @deg,
        @sal
    )
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH

```

## 2.1.2.7. Procedure: InsertNewStd

### Input/Output

	Name	Data type	Description
→@	us_id	int	
→@	dept_id	int	
→@	fname	varchar(50)	
→@	lname	varchar(50)	
→@	bod	date	
→@	address	varchar(50)	

### Uses

	Name
⚙️	InsertNewStd
💻	Student

### Script

```

CREATE PROC InsertNewStd
AS
BEGIN TRY
    INSERT INTO Student
    VALUES (
        @us_id,
        @dept_id,
        @fname,
        @lname,
        @bod,
        @address
    )
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH

```

## 2.1.2.8. Procedure: InsertNewUser

### Input/Output

	Name	Data type	Description
→@	us_name	varchar(50)	
→@	us_mail	varchar(50)	
→@	us_pass	varchar(50)	
→@	us_sex	varchar(1)	
→@	us_isStd	bit	

### Uses

	Name
⚙️	InsertNewUser
>User	

### Script

```

CREATE PROC [dbo].[InsertNewUser]
AS
BEGIN TRY
    INSERT INTO [User]
    VALUES (@us_name, @us_mail, @us_pass, @us_sex, @us_isStd)
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
END CATCH

```

## 2.1.2.9. Procedure: InsertQues

### Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	cont	varchar(200)	
→@	grade	int	
→@	typ	varchar(3)	
→@	ans	int	

### Uses

	Name
⚙️	InsertQues
Question	

### Script

```
CREATE PROC InsertQues  
  
AS  
    BEGIN TRY  
        INSERT INTO Question  
        VALUES (  
            @crs_id,  
            @cont,  
            @grade,  
            @typ,  
            @ans  
        )  
  
    END TRY  
    BEGIN CATCH  
        SELECT  
        ERROR_STATE() AS ErrorState,  
        ERROR_MESSAGE() AS ErrorMessage  
    END CATCH
```

## 2.1.2.10. Procedure: SelectCrsIns

### Input/Output

	Name	Data type	Description
@	ins_id	int	
@	crs_id	int	

### Uses

	Name
⚙️	SelectCrsIns
↳	Ins_Crs

### Script

```
CREATE PROC SelectCrsIns  
  
AS  
    BEGIN TRY  
        SELECT *  
        FROM Ins_Crs  
        WHERE Ins_Id = @ins_id  
        AND Crs_Id = @crs_id  
    END TRY  
    BEGIN CATCH  
        SELECT  
        ERROR_STATE() AS ErrorState,  
        ERROR_MESSAGE() AS ErrorMessage  
    END CATCH  
  
/********************************************************************/
```

### 2.1.2.11. Procedure: SelectInstructor

#### Input/Output

Name	Data type	Description
@ins_id	int	

#### Uses

Name
SelectInstructor
Instructor

#### Script

```
CREATE PROC SelectInstructor @ins_id int
AS
    BEGIN TRY
        SELECT *
        FROM Instructor
        WHERE Ins_Id = @ins_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

## 2.1.2.12. Procedure: SelectStdById

### Input/Output

Name	Data type	Description
@std_id	int	

### Uses

Name
SelectStdById
Student

### Script

```
CREATE PROC SelectStdById
AS
    BEGIN TRY
        SELECT *
        FROM Student
        WHERE Std_Id = @std_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

### 2.1.2.13. Procedure: SelectUserById

#### Input/Output

Name	Data type	Description
@us_id	int	

#### Uses

Name
SelectUserById
User

#### Script

```
CREATE PROC [dbo].[SelectUserById] @us_id int
AS
    BEGIN TRY
        SELECT *
        FROM [User]
        WHERE U_Id = @us_id
    END TRY
    BEGIN CATCH
        SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

## 2.1.2.14. Procedure: sp\_AssignAnsToStd

### Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	exm_id	int	
→@	ans_dict	table type	

### Uses

	Name
⚙️	sp_AssignAnsToStd
grid	Std_Ques_Exm

### Script

```

CREATE PROC sp_AssignAnsToStd
AS
BEGIN TRY
    DECLARE @LOCAL_TABLETABLEVARIABLE TABLE
    (
        std_id int NULL,
        exm_id int NULL,
        ques_id int,
        ans varchar(1)
    )
    INSERT INTO @LOCAL_TABLETABLEVARIABLE (ques_id, ans)
    SELECT * FROM @ans_dict
    UPDATE @LOCAL_TABLETABLEVARIABLE
    SET
        std_id = @std_id,
        exm_id = @exm_id
    INSERT INTO Std_Ques_Exm
    SELECT * FROM @LOCAL_TABLETABLEVARIABLE
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
    
```

## 2.1.2.15. Procedure: sp\_CorrectStdExam

### Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	exm_id	int	

### Uses

	Name
⚙️	sp_CorrectStdExam
📄	Choice
📄	Exam
📄	Question
📄	Std_Crs
📄	Std_Ques_Exm

TRIAL

## Script

```
CREATE PROC [dbo].[sp_CorrectStdExam]
AS
BEGIN TRY
    DECLARE @LOCAL_TABLEVARIABLE TABLE
    (
        mod_ans varchar(1),
        std_ans varchar(1)
    )

    INSERT INTO @LOCAL_TABLEVARIABLE
    SELECT Cho_Char, Std_Answer
    FROM Choice c
    JOIN Question q ON (q.Model_Ans = c.Cho_Id)
    JOIN Std_Ques_Exm sqe ON (sqe.Ques_Id = q.Ques_Id)
    WHERE Exm_Id = @exm_id
    AND Std_Id = @std_id
    declare c cursor
    FOR SELECT * FROM @LOCAL_TABLEVARIABLE

    DECLARE @mod varchar(1), @ans varchar(1)
    DECLARE @scr int, @grade int, @crs_id int
    SET @scr = 0
    SET @grade = 0
    OPEN c
    FETCH c INTO @mod, @ans
    while @@FETCH_STATUS = 0
    BEGIN
        IF @mod = @ans
        BEGIN
            SET @scr = @scr + 1
        END
        FETCH c INTO @mod, @ans
    END
    SELECT @grade = Exm_Grade,
           @crs_id = Crs_Id
    FROM Exam
    WHERE Exm_Id = @exm_id

    DECLARE @ques_grade int
    SET @ques_grade = @grade / 10
    SET @scr = @scr * @ques_grade
    UPDATE Std_Crs
    SET
        Grade = @scr,
        Date = GETDATE()
    WHERE Std_Id = @std_id
    AND Crs_Id = @crs_id
END TRY
BEGIN CATCH
SELECT
    ERROR_STATE() AS ErrorState,
    ERROR_MESSAGE() AS ErrorMessage
END CATCH
```

## 2.1.2.16. Procedure: sp\_CourseNameWithStudentCount

### Input/Output

Name	Data type	Description
@ins_id	int	

### Uses

Name
sp_CourseNameWithStudentCount
Course
Ins_Crs
Std_Crs

### Script

```
CREATE PROC sp_CourseNameWithStudentCount @ins_id INT
as
SELECT Crs_Name,
       COUNT(Std_Id) AS [Students count]
FROM dbo.Course
INNER JOIN dbo.Ins_Crs
    ON dbo.Course.Crs_Id = dbo.Ins_Crs.Crs_Id
INNER JOIN dbo.Std_Crs
    ON Std_Crs.Crs_Id = Course.Crs_Id
WHERE dbo.Ins_Crs.Ins_Id = @ins_id GROUP BY Crs_Name;
```

### 2.1.2.17. Procedure: sp\_deleteChoice

#### Input/Output

Name	Data type	Description
→@ cho_id	int	

#### Uses

Name
sp_deleteChoice
Choice

#### Script

```
CREATE PROC sp_deleteChoice @cho_id INT
AS
BEGIN TRY
    DELETE FROM dbo.Choice
    WHERE Cho_Id = @cho_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.18. Procedure: sp\_deleteCourse

### Input/Output

Name	Data type	Description
→@ crs_id	int	

### Uses

Name
sp_deleteCourse
Course

### Script

```
CREATE PROC sp_deleteCourse @crs_id INT
AS
BEGIN TRY
    DELETE FROM dbo.Course
    WHERE Crs_Id = @crs_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

### 2.1.2.19. Procedure: sp\_deleteDepartment

#### Input/Output

Name	Data type	Description
deptId	int	

#### Uses

Name
sp_deleteDepartment
Department

#### Script

```
CREATE PROC sp_deleteDepartment @deptId INT
AS
BEGIN TRY
    DELETE FROM dbo.Department WHERE Dept_Id=@deptId;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
    ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```

## 2.1.2.20. Procedure: sp\_deleteExam

### Input/Output

Name	Data type	Description
@exam_id	int	

### Uses

Name
sp_deleteExam
Exam

### Script

```
CREATE PROC sp_deleteExam @exam_id INT
AS
BEGIN TRY
    delete FROM dbo.Exam WHERE Exm_Id =@exam_id
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
    ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```

TRIAL

## 2.1.2.21. Procedure: sp\_DeleteQues

### Input/Output

Name	Data type	Description
@ques_id	int	

### Uses

Name
sp_DeleteQues
Question

### Script

```
CREATE PROC sp_DeleteQues
AS
    BEGIN TRY
        DELETE FROM Question
        WHERE Ques_Id = @ques_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
/*
******/
```

## 2.1.2.22. Procedure: sp\_DeleteStdFromCrs

### Input/Output

	Name	Data type	Description
@	std_id	int	
@	crs_id	int	

### Uses

	Name
⚙️	sp_DeleteStdFromCrs
↳	Std_Crs

### Script

```
CREATE PROC sp_DeleteStdFromCrs
    @std_id int,
    @crs_id int
AS
BEGIN TRY
    DELETE FROM Std_Crs
    WHERE Std_Id = @std_id
        AND Crs_Id = @crs_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

### 2.1.2.23. Procedure: sp\_deleteTopic

#### Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	crs_top	varchar(50)	

#### Uses

	Name
⚙️	sp_deleteTopic
↳	Crs_Top

#### Script

```
CREATE PROC sp_deleteTopic
    @crs_id INT,
    @crs_top VARCHAR(50)
AS
BEGIN TRY
    DELETE FROM dbo.Crs_Top
    WHERE Crs_Id = @crs_id
        AND Crs_Top = @crs_top;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.24. Procedure: sp\_ExamQuestionsAndAnswers

### Input/Output

Name	Data type	Description
@exm_id	int	

### Uses

Name
sp_ExamQuestionsAndAnswers
Choice
Exm_Ques
Question

### Script

```
CREATE PROC sp_ExamQuestionsAndAnswers @exm_id INT
as
SELECT Ques_Content, Cho_Content
FROM Exm_Ques eq
INNER JOIN Question q
    ON (q.Ques_Id = eq.Ques_Id)
INNER JOIN Choice c
    ON (c.Ques_Id = q.Ques_Id)
WHERE eq.Exm_Id = @exm_id
```

## 2.1.2.25. Procedure: sp\_GenerateExam

### Input/Output

	Name	Data type	Description
@	mcq	int	
@	tfq	int	
@	crs_id	int	
@	ins_id	int	
@	grade	int	

### Uses

	Name
sp_GenerateExam	
Exam	
Exm_Ques	
Question	

### Script

```

CREATE PROC [dbo].[sp_GenerateExam]
AS
BEGIN TRY
    INSERT INTO Exam
    VALUES (@crs_id, @ins_id, @grade);

    DECLARE @exm_id int;
    SET @exm_id = (SELECT MAX(Exm_Id) FROM Exam);

    DECLARE @LOCAL_TABLEVARIABLE TABLE
    (
        ex_id int NULL,
        ques_id int NULL,
        grade int NULL
    )

    INSERT INTO @LOCAL_TABLEVARIABLE (ques_id)
    SELECT TOP(@mcq) Ques_Id
    FROM Question
    WHERE Ques_Type = 'MCQ' AND Crs_Id = @crs_id
    ORDER BY NEWID()

    INSERT INTO @LOCAL_TABLEVARIABLE (ques_id)
    SELECT TOP(@tfq) Ques_Id
    FROM Question
    WHERE Ques_Type = 'TFQ' AND Crs_Id = @crs_id
    ORDER BY NEWID()

    UPDATE @LOCAL_TABLEVARIABLE
    SET ex_id = @exm_id,
        grade = 1

    INSERT INTO Exm_Ques
    SELECT * FROM @LOCAL_TABLEVARIABLE
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH

```

\*\*\*\*\*

## 2.1.2.26. Procedure: sp\_InsertChoice

### Input/Output

	Name	Data type	Description
→@	ques_id	int	
→@	cho_content	varchar(200)	
→@	cho_char	varchar(1)	

### Uses

	Name
⚙️	sp_InsertChoice
grid	Choice

### Script

```
--choice
CREATE PROC sp_InsertChoice
    @ques_id INT,
    @cho_content VARCHAR(200),
    @cho_char VARCHAR(1)
AS
BEGIN TRY
    INSERT INTO dbo.Choice
    (
        Ques_Id,
        Cho_Content,
        Cho_Char
    )
    VALUES
    (
        @ques_id,      -- Ques_Id - int
        @cho_content, -- Cho_Content - varchar(200)
        @cho_char     -- Cho_Char - varchar(1)
    );
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.27. Procedure: sp\_insertCourse

### Input/Output

	Name	Data type	Description
→@	crs_name	varchar(50)	
→@	crs_desc	varchar(50)	
→@	crs_dur	int	

### Uses

Name
sp_insertCourse
Course

### Script

```
CREATE PROC sp_insertCourse
    @crs_name VARCHAR(50),
    @crs_desc VARCHAR(50),
    @crs_dur INT
AS
BEGIN TRY
    INSERT INTO dbo.Course
    (
        Crs_Name,
        Crs_Desc,
        Crs_Dur
    )
    VALUES
    (
        @crs_name, -- Crs_Name - varchar(50)
        @crs_desc, -- Crs_Desc - varchar(50)
        @crs_dur   -- Crs_Dur - int
    );
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.28. Procedure: sp\_insertDepartment

### Input/Output

	Name	Data type	Description
→@	dept_name	varchar(50)	
→@	dept_loc	varchar(50)	
→@	dept_managerHireDate	date	
→@	dept_managerId	int	

### Uses

Name
sp_insertDepartment
Department

### Script

```
CREATE PROC sp_insertDepartment
    @dept_name VARCHAR(50),
    @dept_loc VARCHAR(50),
    @dept_managerHireDate DATE,
    @dept_managerId INT
AS
BEGIN TRY
    INSERT INTO dbo.Department
    (
        Dept_Name,
        Dept_Loc,
        Dept_ManagerHireDate,
        Dept_ManagerId
    )
    VALUES
    (
        @dept_name,          -- Dept_Name - varchar(50)
        @dept_loc,           -- Dept_Loc - varchar(50)
        @dept_managerHireDate, -- Dept_ManagerHireDate - date
        @dept_managerId      -- Dept_ManagerId - int
    );
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.29. Procedure: sp\_insertExam

### Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	generator_id	int	
→@	exam_grade	int	

### Uses

	Name
⚙️	sp_insertExam
📝	Exam

### Script

```
--exam
CREATE PROC sp_insertExam @crs_id INT,@generator_id INT ,@exam_grade INT
AS
BEGIN TRY
    INSERT INTO dbo.Exam
    (
        Crs_Id,
        Generator_Id,
        Exm_Grade
    )
    VALUES
    (
        @crs_id, -- Crs_Id - int
        @generator_id, -- Generator_Id - int
        @exam_grade -- Exm_Grade - int
    )
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```

### 2.1.2.30. Procedure: sp\_InsertStdAnsForExm

#### Input/Output

	Name	Data type	Description
@	std_id	int	
@	Exm_id	int	
@	ques_id	int	
@	ans	varchar(1)	

#### Uses

Name
sp_InsertStdAnsForExm
Std_Ques_Exm

#### Script

```
*****  
CREATE PROC sp_InsertStdAnsForExm  
  
AS  
    BEGIN TRY  
        INSERT INTO Std_Ques_Exm  
        VALUES (  
            @std_id ,  
            @Exm_id ,  
            @ques_id ,  
            @ans  
        )  
    END TRY  
    BEGIN CATCH  
        SELECT  
        ERROR_STATE() AS ErrorState,  
        ERROR_MESSAGE() AS ErrorMessage  
    END CATCH
```

### 2.1.2.31. Procedure: sp\_InsertStdForCrs

#### Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	crs_id	int	
→@	grade	int	
→@	date	date	

#### Uses

Name
sp_InsertStdForCrs
Std_Crs

#### Script

```
CREATE PROC sp_InsertStdForCrs
AS
BEGIN TRY
    INSERT INTO Std_Crs
    VALUES (
        @std_id,
        @crs_id,
        @grade,
        @date
    )
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

### 2.1.2.32. Procedure: sp\_insertTopic

#### Input/Output

	Name	Data type	Description
@	crs_id	int	
@	crs_top	varchar(50)	

#### Uses

	Name
⚙️	sp_insertTopic
↳	Crs_Top

#### Script

```
CREATE PROC sp_insertTopic
    @crs_id INT,
    @crs_top VARCHAR(50)
AS
BEGIN TRY
    INSERT INTO dbo.Crs_Top
    (
        Crs_Id,
        Crs_Top
    )
    VALUES
    (
        @crs_id, -- Crs_Id - int
        @crs_top -- Crs_Top - varchar(50)
    );
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

### 2.1.2.33. Procedure: sp\_selectchoice

#### Input/Output

Name	Data type	Description
→@ id	int	

#### Uses

Name
sp_selectchoice
Choice

#### Script

```
CREATE PROC sp_selectchoice @id INT
AS
BEGIN TRY
    SELECT *
    FROM dbo.Choice
    WHERE Cho_Id = @id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;

---course
```

TRIAL

### 2.1.2.34. Procedure: sp\_selectCourse

#### Input/Output

Name	Data type	Description
→@ crs_id	int	

#### Uses

Name
sp_selectCourse
Course

#### Script

```
CREATE PROC sp_selectCourse @crs_id INT
AS
BEGIN TRY
    SELECT *
    FROM dbo.Course
    WHERE Crs_Id = @crs_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;

--crs_top
```

### 2.1.2.35. Procedure: sp\_selectDepartment

#### Input/Output

Name	Data type	Description
@dept_id	int	

#### Uses

Name
sp_selectDepartment
Department

#### Script

```
CREATE PROC sp_selectDepartment @dept_id INT
AS
BEGIN TRY
    SELECT * FROM dbo.Department WHERE Dept_Id =@dept_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
    ERROR_MESSAGE() AS ErrorMessag;
END CATCH
```

TRIAL

### 2.1.2.36. Procedure: sp\_selectExam

#### Input/Output

Name	Data type	Description
exam_id	int	

#### Uses

Name
sp_selectExam
exam

#### Script

```
CREATE PROC sp_selectExam @exam_id INT
AS
BEGIN TRY
    SELECT * FROM exam WHERE Exam_Id =@exam_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
    ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```

TRIAL

## 2.1.2.37. Procedure: sp\_SelectQues

### Input/Output

Name	Data type	Description
@ques_id	int	

### Uses

Name
sp_SelectQues
Question

### Script

```
CREATE PROC sp_SelectQues
AS
    BEGIN TRY
        SELECT *
        FROM Question
        WHERE Ques_Id = @ques_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

### 2.1.2.38. Procedure: sp\_SelectStdAnsForExm

#### Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	Exm_id	int	
→@	ques_id	int	

#### Uses

Name
sp_SelectStdAnsForExm
Std_Ques_Exm

#### Script

```
CREATE PROC sp_SelectStdAnsForExm
    @std_id int,
    @Exm_id int,
    @ques_id int
AS
BEGIN TRY
    SELECT *
    FROM Std_Ques_Exm
    WHERE Std_Id = @std_id
        AND Exm_Id = @Exm_id
        AND Ques_Id = @ques_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

### 2.1.2.39. Procedure: sp\_SelectStdOfCrs

#### Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	crs_id	int	

#### Uses

	Name
⚙️	sp_SelectStdOfCrs
↳	Std_Crs

#### Script

```
CREATE PROC sp_SelectStdOfCrs
    @std_id int,
    @crs_id int
AS
BEGIN TRY
    SELECT *
    FROM Std_Crs
    WHERE Std_Id = @std_id
        AND Crs_Id = @crs_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

#### 2.1.2.40. Procedure: sp\_selectTopic

##### Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	crs_topic	varchar(50)	

##### Uses

	Name
⚙️	sp_selectTopic
↳	Crs_Top

##### Script

```
CREATE PROC sp_selectTopic
    @crs_id INT,
    @crs_topic VARCHAR(50)
AS
BEGIN TRY
    SELECT *
    FROM dbo.Crs_Top
    WHERE Crs_Id = @crs_id
        AND Crs_Top = @crs_topic;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;

--dept
```

#### 2.1.2.41. Procedure: sp\_StdExamAnswers

##### Input/Output

	Name	Data type	Description
@	exm_id	int	
@	std_id	int	

##### Uses

	Name
⚙️	sp_StdExamAnswers
↳	Choice
↳	Question
↳	Std_Ques_Exm

##### Script

```
CREATE PROC sp_StdExamAnswers @exm_id INT, @std_id INT
as
SELECT Ques_Content, Std_Answer
FROM Std_Ques_Exm sqe
INNER JOIN Question q
    ON (q.Ques_Id = sqe.Ques_Id)
INNER JOIN Choice c
    ON (sqe.Std_Answer = c.Cho_Id)
WHERE sqe.Exm_Id = @exm_id
    AND sqe.Std_Id = @std_id
```

## 2.1.2.42. Procedure: sp\_StudentGradePerCourse

### Input/Output

	Name	Data type	Description
→@	std_id	int	

### Uses

	Name
⚙️	sp_StudentGradePerCourse
\grid	Course
\grid	Std_Crs
\grid	Student

### Script

```
CREATE PROC sp_StudentGradePerCourse @std_id INT
AS
SELECT CONCAT(Std_Fname, ' ', Std_Lname) AS Student, c.Crs_Name AS Course, grade AS Grade
FROM Std_Crs sc
JOIN Student s
ON (s.Std_Id = sc.Std_Id)
JOIN Course c
ON (c.Crs_Id = sc.Crs_Id)
WHERE sc.Std_Id = @std_id
AND sc.Grade IS NOT NULL
```

### 2.1.2.43. Procedure: sp\_StudentperDepartment

#### Input/Output

Name	Data type	Description
→@ dept_id	int	

#### Uses

Name
⚙️ sp_StudentperDepartment
תלמיד Student

#### Script

```
CREATE PROC sp_StudentperDepartment @dept_id INT  
as  
SELECT * FROM dbo.Student s WHERE s.Dept_Id = @dept_id;
```

TRIAL

#### 2.1.2.44. Procedure: sp\_TopicsPerCourse

##### Input/Output

Name	Data type	Description
→@ crs_id	int	

##### Uses

Name
⚙️ sp_TopicsPerCourse
📊 Crs_Top

##### Script

```
CREATE PROC sp_TopicsPerCourse @crs_id INT  
as  
SELECT dbo.Crs_Top.Crs_Top AS Topics FROM dbo.Crs_Top WHERE crs_id =@crs_id
```

TRIAL

#### 2.1.2.45. Procedure: sp\_updateChoice

##### Input/Output

	Name	Data type	Description
→@	cho_id	int	
→@	ques_id	int	
→@	cho_content	varchar(200)	
→@	cho_char	varchar(1)	

##### Uses

Name
sp_updateChoice
Choice

##### Script

```
CREATE PROC sp_updateChoice
    @cho_id INT,
    @ques_id INT,
    @cho_content VARCHAR(200),
    @cho_char VARCHAR(1)
AS
BEGIN TRY
    UPDATE dbo.Choice
    SET Ques_Id = @ques_id,
        Cho_Content = @cho_content,
        Cho_Char = @cho_char
    WHERE Cho_Id = @cho_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

#### 2.1.2.46. Procedure: sp\_updateCourse

##### Input/Output

	Name	Data type	Description
→@	crs_id	int	
→@	crs_name	varchar(50)	
→@	crs_desc	varchar(50)	
→@	crs_dur	int	

##### Uses

Name
sp_updateCourse
Course

##### Script

```
CREATE PROC sp_updateCourse
    @crs_id INT,
    @crs_name VARCHAR(50),
    @crs_desc VARCHAR(50),
    @crs_dur INT
AS
BEGIN TRY
    UPDATE dbo.Course
    SET Crs_Name = @crs_name,
        Crs_Desc = @crs_desc,
        Crs_Dur = @crs_dur
    WHERE Crs_Id = @crs_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.47. Procedure: sp\_UpdateCrsForIns

### Input/Output

	Name	Data type	Description
→@	ins_id	int	
→@	old_crs_id	int	
→@	new_crs_id	int	

### Uses

	Name
⚙️	sp_UpdateCrsForIns
grid	Ins_Crs

### Script

```
CREATE PROC sp_UpdateCrsForIns
AS
    BEGIN TRY
        UPDATE Ins_Crs
        SET
            Crs_Id = @new_crs_id
        WHERE Crs_Id = @ins_id
        AND Crs_Id = @old_crs_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

## 2.1.2.48. Procedure: sp\_UpdateCrsOfStd

### Input/Output

	Name	Data type	Description
→@	std_id	int	
→@	old_crs_id	int	
→@	new_crs_id	int	
→@	grade	int	
→@	date	date	

### Uses

	Name
⚙️	sp_UpdateCrsOfStd
💻	Std_Crs

### Script

```
CREATE PROC sp_UpdateCrsOfStd
```

```
AS
```

```
    BEGIN TRY
        UPDATE Std_Crs
        SET
            Crs_Id = @new_crs_id,
            Grade = @grade,
            [Date] = @date
        WHERE Std_Id = @std_id
        AND Crs_Id = @old_crs_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

```
@std_id int,  
@old_crs_id int,  
@new_crs_id int,  
@grade int = NULL,  
@date date = NULL
```

## 2.1.2.49. Procedure: sp\_updateDepartment

### Input/Output

	Name	Data type	Description
→@	dept_id	int	
→@	dept_name	varchar(50)	
→@	dept_loc	varchar(50)	
→@	dept_managerHireDate	date	
→@	dept_managerId	int	

### Uses

	Name
⚙️	sp_updateDepartment
💻	Department

### Script

```
CREATE PROC sp_updateDepartment
    @dept_id INT,
    @dept_name VARCHAR(50),
    @dept_loc VARCHAR(50),
    @dept_managerHireDate DATE,
    @dept_managerId INT
AS
BEGIN TRY
    UPDATE dbo.Department
    SET Dept_Name = @dept_name,
        Dept_Loc = @dept_loc,
        Dept_ManagerHireDate = @dept_managerHireDate,
        Dept_ManagerId = @dept_managerId
    WHERE Dept_Id = @dept_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.50. Procedure: sp\_updateExam

### Input/Output

	Name	Data type	Description
→@	exam_id	int	
→@	crs_id	int	
→@	generator_id	int	
→@	exam_grade	int	

### Uses

Name
sp_updateExam
Exam

### Script

```
CREATE PROC sp_updateExam @exam_id INT ,@crs_id INT ,@generator_id INT ,@exam_grade INT
AS
BEGIN TRY
    UPDATE dbo.Exam SET Crs_Id =@crs_id,Generator_Id =@generator_id,Exm_Grade=@exam_grade
    WHERE Exm_Id =@exam_id;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```

## 2.1.2.51. Procedure: sp\_updateTopic

### Input/Output

	Name	Data type	Description
@	crs_id	int	
@	crs_top	varchar(50)	

### Uses

	Name
⚙️	sp_updateTopic
↳	Crs_Top

### Script

```
CREATE PROC sp_updateTopic
    @crs_id INT,
    @crs_top VARCHAR(50)
AS
BEGIN TRY
    UPDATE dbo.Crs_Top
    SET Crs_Top = @crs_top
    WHERE Crs_Id = @crs_id
        AND Crs_Top = @crs_top;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

## 2.1.2.52. Procedure: UpdateInstructor

### Input/Output

	Name	Data type	Description
@	ins_id	int	
@	u_id	int	
@	dept_id	int	
@	fname	varchar(50)	
@	lname	varchar(50)	
@	deg	varchar(50)	
@	sal	int	

### Uses

Name
UpdateInstructor
Instructor

### Script

```
CREATE PROC UpdateInstructor
```

```
AS
BEGIN TRY
    UPDATE Instructor
    SET
        U_Id = @u_id,
        Dept_Id = @dept_id,
        Ins_Fname = @fname,
        Ins_Lname = @lname,
        Ins_Degree = @deg,
        Ins_Salary = @sal
    WHERE Ins_Id = @ins_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH

```

\*\*\*\*\*

### 2.1.2.53. Procedure: UpdateQues

#### Input/Output

	Name	Data type	Description
→@	ques_id	int	
→@	crs_id	int	
→@	cont	varchar(200)	
→@	grade	int	
→@	typ	varchar(3)	
→@	ans	int	

#### Uses

	Name
⚙️	UpdateQues
📄	Question

#### Script

```
CREATE PROC UpdateQues
AS
BEGIN TRY
    UPDATE Question
    SET
        Crs_Id = @crs_id,
        Ques_Content = @cont,
        Ques_Grade = @grade,
        Ques_Type = @typ,
        Model_Ans = @ans
    WHERE Ques_Id = @ques_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH
```

## 2.1.2.54. Procedure: UpdateStdInfo

### Input/Output

	Name	Data type	Description
@	std_id	int	
@	us_id	int	
@	dept_id	int	
@	fname	varchar(50)	
@	lname	varchar(50)	
@	bod	date	
@	address	varchar(50)	

### Uses

Name
UpdateStdInfo
Student

### Script

```

CREATE PROC UpdateStdInfo
AS
BEGIN TRY
    UPDATE Student
    SET
        U_Id = @us_id,
        Dept_Id = @dept_id,
        Std_Fname = @fname,
        Std_Lname = @lname,
        Std_BOD = @bod,
        Std_Address = @address
    WHERE Std_Id = @std_id
END TRY
BEGIN CATCH
    SELECT
        ERROR_STATE() AS ErrorState,
        ERROR_MESSAGE() AS ErrorMessage
    END CATCH

```

## 2.1.2.55. Procedure: UpdateUser

### Input/Output

	Name	Data type	Description
→@	us_id	int	
→@	us_name	varchar(50)	
→@	us_mail	varchar(50)	
→@	us_pass	varchar(50)	
→@	us_sex	varchar(1)	
→@	us_isStd	bit	

### Uses

	Name
⚙️	UpdateUser
>User	

### Script

```

CREATE PROC [dbo].[UpdateUser]
AS
    BEGIN TRY
        UPDATE [User]
        SET
            U_UserName = @us_name,
            U_Email = @us_mail,
            U_Password = @us_pass,
            U_Sex = @us_sex,
            U_IsStd = @us_isStd
        WHERE U_Id = @us_id
    END TRY
    BEGIN CATCH
        SELECT
            ERROR_STATE() AS ErrorState,
            ERROR_MESSAGE() AS ErrorMessage
    END CATCH

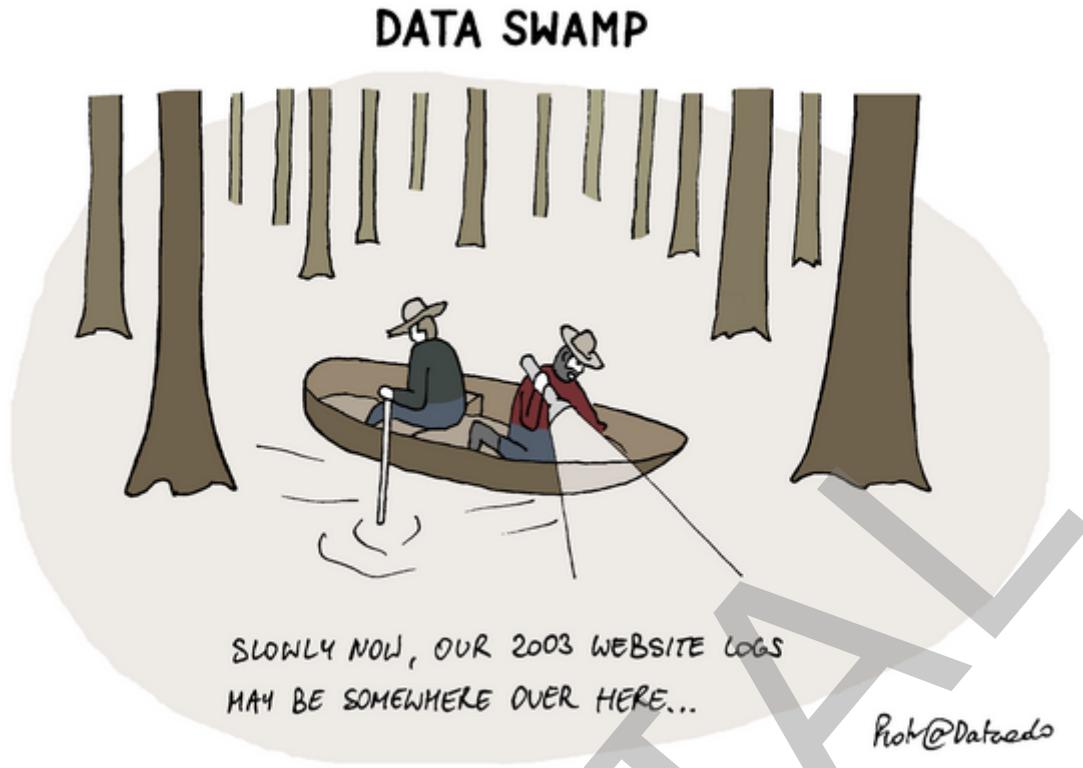
```

TRIAL

### 3. Sample Data Lake

This is a documentation of files in sample data lake.

And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

## 3.1. US Census - 2018 Business Dynamics

### 3.1.1. Structures

#### 3.1.1.1. Structure: bds2018\_cty.csv (Business Dynamics 2018 - City)

Business Dynamics Statistics in 2018 classified by city.

##### Columns

	Name	Data type	Description / Attributes
»	year	Int	Year <b>Nullable</b>
»	st (State)	Int	State <b>Nullable</b>
»	cty (City)	Int	City <b>Nullable</b>
»	firms (Firms)	String	Number of firms that exited during the last 12 months <b>Nullable</b>
»	estabs (Establishments)	String	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
»	emp (Employees)	String	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
»	denom ((DHS) denominator)	String	(DHS) denominator <b>Nullable</b>
»	estabs_entry	String	Number of establishments born during the last 12 months <b>Nullable</b>
»	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
»	estabs_exit	String	Number of establishments exited during the last 12 months <b>Nullable</b>
»	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
»	job_creation	String	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
»	job_creation_births	String	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
»	job_creation_continuers	String	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
»	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
»	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
»	job_destruction	String	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_deaths	String	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
	job_destruction_continuers	String	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	String	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIA

### 3.1.1.2. Structure: bds2018\_eage.csv (Business Dynamics 2018 - Establishment Age)

Business Dynamics Statistics in 2018 classified by establishment age.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	eage	String	Establishment age <b>Nullable</b>
▀	firms (Firms)	String	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	String	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	String	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	String	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	String	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	String	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	String	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	String	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	String	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	String	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	String	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	Int	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.3. Structure: bds2018\_esize.csv (Business Dynamics 2018 - Establishment Size)

Business Dynamics Statistics in 2018 classified by establishment size.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	esize	String	Establishment size <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	String	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.4. Structure: bds2018\_fage.csv (Business Dynamics 2018 - Firm Age)

Business Dynamics Statistics in 2018 classified by firm age.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	fage	String	Firm age <b>Nullable</b>
▀	firms (Firms)	String	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	String	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	String	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	String	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	String	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	String	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	String	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	String	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	String	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	String	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	String	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.5. Structure: bds2018\_fsize.csv (Business Dynamics 2018 - Firm Size)

Business Dynamics Statistics in 2018 classified by firm size.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	fsize	String	Firm size <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	Int	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.6. Structure: bds2018\_iesize.csv (Business Dynamics 2018 - Initial Establishment Size)

Business Dynamics Statistics in 2018 classified by initial establishment size.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	iesize	String	Initial establishment size <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	String	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	Int	<b>Nullable</b>
	firmdeath_estabs	Int	<b>Nullable</b>
	firmdeath_emp	Int	<b>Nullable</b>

TRIAL

### 3.1.1.7. Structure: bds2018\_ifsize.csv (Business Dynamics 2018 - Initial Firm Size)

Business Dynamics Statistics in 2018 classified by initial firm size.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	ifsize	String	Initial firm size <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	Int	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.8. Structure: bds2018\_metro.csv (Business Dynamics 2018 - Metro/Non-Metro)

Business Dynamics Statistics in 2018 classified by Metro/Non-metro.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	metro	String	Metro/Non-Metro <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	Int	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	DateTime	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	Int	<b>Nullable</b>
	firmdeath_estabs	Int	<b>Nullable</b>
	firmdeath_emp	Int	<b>Nullable</b>

TRIAL

### 3.1.1.9. Structure: bds2018\_sector.csv (Business Dynamics 2018 - Sector)

Business Dynamics Statistics in 2018 classified by sector.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	sector	String	Sector <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	String	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	String	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	String	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	String	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	String	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	String	<b>Nullable</b>
	firmdeath_estabs	String	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.10. Structure: bds2018\_st.csv (Business Dynamics 2018 - State)

Business Dynamics Statistics in 2018 classified by state.

#### Columns

	Name	Data type	Description / Attributes
▀	year	Int	Year <b>Nullable</b>
▀	st	Int	State <b>Nullable</b>
▀	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
▀	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
▀	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
▀	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
▀	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
▀	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
▀	estabs_exit_rate	String	Rate of establishments exited during the last 12 months <b>Nullable</b>
▀	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate_births	String	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
▀	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
▀	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_continuers	String	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
▀	job_destruction_rate_deaths	String	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	Int	<b>Nullable</b>
	firmdeath_estabs	Int	<b>Nullable</b>
	firmdeath_emp	String	<b>Nullable</b>

TRIAL

### 3.1.1.11. Structure: bds2018.csv (Business Dynamics 2018 - Economy-wide)

Business Dynamics Statistics in 2018 for entire economy.

#### Columns

	Name	Data type	Description / Attributes
»	year	Int	Year <b>Nullable</b>
»	firms (Firms)	Int	Number of firms that exited during the last 12 months <b>Nullable</b>
»	estabs (Establishments)	Int	Number of establishments associated with firm deaths during the last 12 months <b>Nullable</b>
»	emp (Employees)	Int	Number of employees associated with firm deaths during the last 12 months <b>Nullable</b>
»	denom ((DHS) denominator)	Int	(DHS) denominator <b>Nullable</b>
»	estabs_entry	Int	Number of establishments born during the last 12 months <b>Nullable</b>
»	estabs_entry_rate	String	Rate of establishments born during the last 12 months <b>Nullable</b>
»	estabs_exit	Int	Number of establishments exited during the last 12 months <b>Nullable</b>
»	estabs_exit_rate	DateTime	Rate of establishments exited during the last 12 months <b>Nullable</b>
»	job_creation	Int	Number of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
»	job_creation_births	Int	Number of jobs created from opening establishments during the last 12 months <b>Nullable</b>
»	job_creation_continuers	Int	Number of jobs created from expanding establishments during the last 12 months <b>Nullable</b>
»	job_creation_rate_births	DateTime	Rate of jobs created from opening establishments during the last 12 months <b>Nullable</b>
»	job_creation_rate	String	Rate of jobs created from expanding and opening establishments during the last 12 months <b>Nullable</b>
»	job_destruction	Int	Number of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
»	job_destruction_deaths	Int	Number of jobs lost from closing establishments during the last 12 months <b>Nullable</b>
»	job_destruction_continuers	Int	Number of jobs lost from contracting establishments during the last 12 months <b>Nullable</b>
»	job_destruction_rate_deaths	DateTime	Rate of jobs lost from closing establishments during the last 12 months <b>Nullable</b>

	job_destruction_rate	String	Rate of jobs lost from contracting and closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation	Int	Number of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	net_job_creation_rate	String	Rate of net jobs created from expanding/contracting and opening/closing establishments during the last 12 months <b>Nullable</b>
	reallocation_rate	String	Rate of reallocation during the last 12 months <b>Nullable</b>
	firmdeath_firms	Int	<b>Nullable</b>
	firmdeath_estabs	Int	<b>Nullable</b>
	firmdeath_emp	Int	<b>Nullable</b>

TRIAL

TRIAL

## 3.2. Fortune 500

### 3.2.1. Structures

#### 3.2.1.1. Structure: 2015Fortune500.csv (Fortune 500 Companies (2015))

Data set of Fortune 500 companies in 2015.

##### Columns

Name		Data type	Description / Attributes
2015_rank	Int	2015 company Fortune 500 rank <b>Nullable</b>	
Company	String	Company name <b>Nullable</b>	
2015_revenues_m	Int	Revenue in millions USD <b>Nullable</b>	

##### Unique keys

Columns	Name / Description
2015_rank	pk_2015Fortune500.csv_2015_rank

### 3.2.1.2. Structure: Fortune500.csv (Fortune 500 Companies (2016))

Data set of Fortune 500 companies in 2016.

#### Columns

		Name	Data type	Description / Attributes
»	 RANK		Int	Company Fortune 500 rank <b>Nullable</b>
»	COMPANY		String	Company name <b>Nullable</b>
»	SYMBOL		String	Ticker symbol <b>Nullable</b> References: company_tickers.json (Company tickers)
»	REVENUES(\$M)		Int	2016 annual revenues in millions USD <b>Nullable</b>
»	WEBSITE		String	Company website address <b>Nullable</b>

#### Links to

Table		Join	Title / Name / Description
 company_tickers.json (Company tickers)		Fortune500.csv SYMBOL = company_tickers.json.N.ticker	fk_company_tickers_json_Fortune500_csv

#### Unique keys

Columns		Name / Description
 RANK		pk_Fortune500.csv_RANK

#### Uses

Name	
 Fortune500.csv (Fortune 500 Companies (2016))	
 company_tickers.json (Company tickers)	

TRIAL

## 3.3. Geography

### 3.3.1. Structures

#### 3.3.1.1. Structure: zips.json (Zips)

##### Columns

Name		Data type	Description / Attributes
[-]	city (City name)	String	holds the city name. A city can have more than one zip code associated with it as different sections of the city can each have a different zip code <b>Nullable</b>
[ ]	loc (Location)	Double[]	holds the location as a longitude latitude pair <b>Nullable</b>
[-]	pop (Population)	Int32	holds the population <b>Nullable</b>
[-]	state (State)	String	holds the two letter state abbreviation <b>Nullable</b>
[-]	🔑 _id (Zip)	String	holds the zip code as a string <b>Nullable</b>

##### Unique keys

Columns		Name / Description
🔑	_id	pk_zips.json_id

### 3.3.1.2. Structure: country.xml (Countries)

List of countries from World Bank.

Columns

	Name	Data type	Description / Attributes
{}	wb:countries	Document	Root element <b>Nullable</b>
[ ]	wb:countries.@page	Int32	<b>Nullable</b>
[ ]	wb:countries.@pages	Int32	<b>Nullable</b>
[ ]	wb:countries.@per_page	Int32	<b>Nullable</b>
[ ]	wb:countries.@total	Int32	<b>Nullable</b>
[ ]	wb:countries.@xmlns:wb	String	<b>Nullable</b>
[{}]	wb:countries.wb:country	Document[]	Country <b>Nullable</b>
[ ]	wb:countries.wb:country.@id	String	3 digit code, e.g. "USA" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:iso2Code (Country ISO 2 code)	String	Country ISO 2 code <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:name (Country name)	String	Country name, e.g. "United States" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:region (Region)	String	Region, e.g. "North America" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:region. @id	String	Region 3 digit code, e.g. "NAC" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:region. @iso2code	String	Region ISO 2 Code, e.g. "XU" <b>Nullable</b>
(...)	wb:countries.wb:country.wb:adminr egion	Document/String	<b>Nullable</b>
[ ]	wb:countries.wb:country.wb:adminr egion.@id	String	<b>Nullable</b>
[ ]	wb:countries.wb:country.wb:adminr egion. @iso2code	String	<b>Nullable</b>
[ ]	wb:countries.wb:country.wb:income Level (Income Level)	String	Income Level, e.g. "High income" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:income Level. @id	String	Income Level 3 digit code, e.g. "HIC" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:income Level. @iso2code	String	Income Level ISO 2 Code, e.g. "XD" <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:lending Type (Lending Type)	String	Lending Type <b>Nullable</b>
[ ]	wb:countries.wb:country.wb:lending Type. @id	String	<b>Nullable</b>

<input type="checkbox"/>	wb:countries.wb:country.wb:lendingType.@iso2code	String	<b>Nullable</b>
<input type="checkbox"/>	wb:countries.wb:country.wb:capitalCity (Capital City)	String	Capital City <b>Nullable</b>
<input type="checkbox"/>	wb:countries.wb:country.wb:longitude (Longitude)	String	Longitude <b>Nullable</b>
<input type="checkbox"/>	wb:countries.wb:country.wb:latitude (Latitude)	String	Latitude <b>Nullable</b>

TRIAL

TRIAL

## 3.4. U.S. Securities and Exchange Commission

### 3.4.1. Structures

#### 3.4.1.1. Structure: company\_tickers.json (Company tickers)

Company tickers from U.S. Securities and Exchange Commission.

##### Columns

		Name	Data type	Description / Attributes
{}	N	Document		Document that holds information about specific company <b>Nullable</b>
└─	N.cik_str (Central Index Key (CIK))	Int32		The Central Index Key (CIK) is used on the SEC's computer systems to identify corporations and individual people who have filed disclosure with the SEC. <b>Nullable</b>
└─	N.ticker (Company ticker)	String		Company ticker, e.g. "AAPL" <b>Nullable</b>
└─	N.title (Company name)	String		Company name, e.g. "Apple Inc." <b>Nullable</b>

##### Linked from

Table		Join	Title / Name / Description
→ Fortune500.csv (Fortune 500 Companies (2016))	company_tickers.json	N.ticker = Fortune500.csvSYMBOL	fk_company_tickers_json_Fortune500_csv

##### Used By

Name	
company_tickers.json	(Company tickers)
→ Fortune500.csv	(Fortune 500 Companies (2016))

TRIAL

## 3.5. HDFS formats

### 3.5.1. Structures

#### 3.5.1.1. Structure: users.avro.[example.avro.User]

Sample Avro file with one (example.avro.User) record.

##### Columns

Name		Data type	Description / Attributes
	name	string	User's name.
	favorite_color	[string, null]	User's favorite color. <b>Nullable</b>
	favorite_numbers	int[]	List of user's favorite numbers.

TRIAL

### 3.5.1.2. Structure: users.orc

Sample ORC file.

#### Columns

	Name	Data type	Description / Attributes
1	name	String	User's name. <b>Nullable</b>
2	favorite_color	String	User's favorite color. <b>Nullable</b>
3	favorite_numbers	Int[]	List of user's favorite numbers. <b>Nullable</b>

TRIAL

### 3.5.1.3. Structure: users.parquet

Sample Parquet file.

Columns

	Name	Data type	Description / Attributes
1	name	String	User's name.
2	favorite_color	String	User's favorite color. <b>Nullable</b>
3	favorite_numbers	Int32[]	List of user's favorite numbers. <b>Nullable</b>

TRIAL

TRIAL

## 3.6. Insurance (Excel)

### 3.6.1. Structures

#### 3.6.1.1. Structure: insurance\_data.xlsx.tblPolicies (Insurance Data)

Sample excel table with fake insurance data.

Columns

	Name	Data type	Description / Attributes
1	Policy	Int	Policy ID number
2	Expiry	Date	Policy expire date.
3	Location	String	Location area: Urban/Rural
4	State	String	U.S. state code.
5	Region	String	U.S. geographic region (Central, East, Northwest etc.)
6	InsuredValue	Int	Value of property covered in a policy
7	Construction	String	Construction material/type (masonry, frame, fire resist etc.)
8	BusinessType	String	Type of business ran in covered building
9	Earthquake	String	If disaster was an earthquake (Y/N)
10	Flood	String	If disaster was a flood (Y/N)

TRIAL

## 3.7. Employees (Delta Lake)

### 3.7.1. Structures

#### 3.7.1.1. Structure: employees.Delta (Emloyees)

Sample Delta Lake file with employees data.

##### Columns

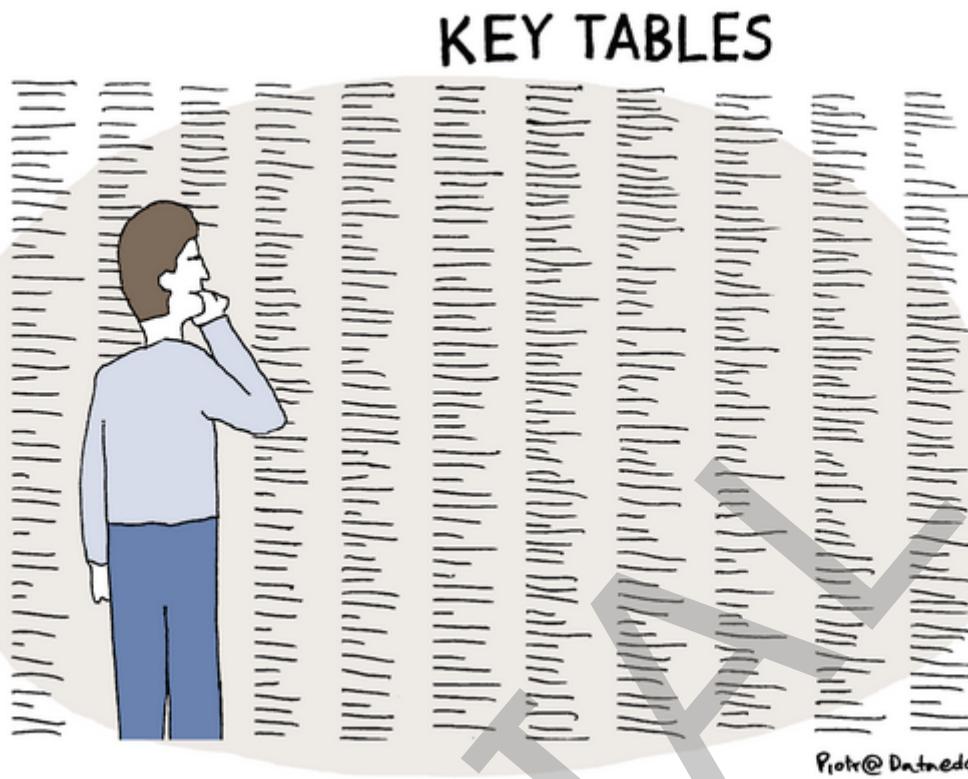
	Name	Data type	Description / Attributes
1	id	Int64	ID of an employee. <b>Nullable</b>
2	first_name	String	Employee's first name. <b>Nullable</b>
3	last_name	String	Employee's last name. <b>Nullable</b>
4	salary	Double	Employee's salary. <b>Nullable</b>
5	employment_date	DateTimeOffset	Date when employee was employed. <b>Nullable</b>

TRIAL

## 4. Sample Data Profiling Database

This is a sample of data profiling of relational database.

And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

## 4.1. Data Dictionary

### 4.1.1. Tables

#### 4.1.1.1. Table: Clients (Clients)

List of all Clients.

Columns

	Name	Data type	Description / Attributes
█	ClientID 	int	Row ID <b>Identity / Auto increment</b>
█	Name	varchar(255)	First name <b>Nullable</b>
█	LastName	varchar(255)	Last name <b>Nullable</b>
█	Email	varchar(255)	Registration email <b>Nullable</b>
█	PhoneNumber	varchar(100)	Phone number with area code <b>Nullable</b>
█	Address	varchar(255)	Mailing address <b>Nullable</b>
█	City	varchar(255)	Mailing address City <b>Nullable</b>
█	Country	varchar(100)	Country (full name) <b>Nullable</b>

Linked from

Table	Join	Title / Name / Description
← Orders (Placed orders)	Clients.ClientID = Orders.ClientID	fk_Clients_Orders

Unique keys

Columns	Name / Description
█ ClientID 	PK_Clients_6EDBB77240EA1FB9

Used By

Name
█ Clients (Clients)
→ Orders (Placed orders)

#### 4.1.1.2. Table: Items (Items in stock)

List of all Items in stock.

Columns

		Name	Data type	Description / Attributes
grid	key	ItemID	int	Row ID <b>Identity / Auto increment</b>
grid		Brand	varchar(255)	Producer of the Item <b>Nullable</b>
grid		Model	varchar(255)	Model name <b>Nullable</b>
grid	key	Code	uniqueidentifier	Item code <b>Nullable</b>
grid		Price	int	Retail price in dollars <b>Nullable</b>

Linked from

Table		Join	Title / Name / Description
grid	Orders (Placed orders)	ItemsItemID = OrdersItemID	fk_Items_Orders

Unique keys

Columns		Name / Description
key	ItemID	PK_Items_727E83EB1C408F82
key	Code	uk_Items_Code

Used By

Name	
grid	Items (Items in stock)
grid	Orders (Placed orders)

### 4.1.1.3. Table: Orders (Placed orders)

List of all Items ordered by Clients.

Columns

		Name	Data type	Description / Attributes
grid	key	OrderID	int	Row ID <b>Identity / Auto increment</b>
grid		ClientID	int	Ref: Clients <b>Nullable</b> References: Clients (Clients)
grid		Description	varchar(MAX)	Comment from Client <b>Nullable</b>
grid		Amount	int	Amount ordered <b>Nullable</b>
grid		ItemID	int	Ref: Items <b>Nullable</b> References: Items (Items in stock)
grid		Date	date	Ordered date <b>Nullable</b>

Links to

Table		Join	Title / Name / Description
grid	Clients (Clients)	OrdersClientID = ClientsClientID	fk_Clients_Orders
grid	Items (Items in stock)	OrdersItemID = ItemsItemID	fk_Items_Orders

Unique keys

Columns		Name / Description
key	OrderID	PK_Orders_630B99565460E9B8

Uses

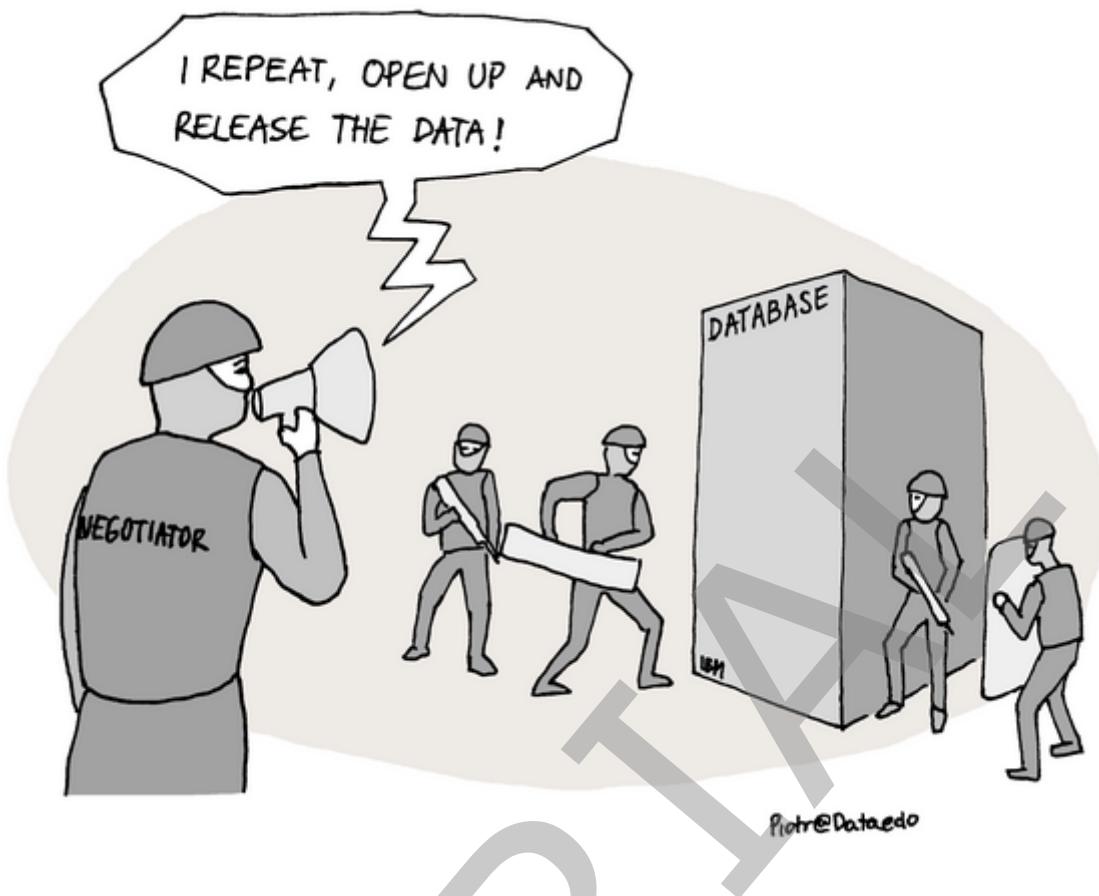
Name	
grid	Orders (Placed orders)
grid	Clients (Clients)
grid	Items (Items in stock)

TRIAL

## 5. Sample Elasticsearch Database (Search Engine)

This is a documentation of a sample **Elasticsearch** database.

And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

## 5.1. Data Dictionary

### 5.1.1. Tables

#### 5.1.1.1. Search index: bank

Columns

ID	Name	Data type	Description / Attributes
	_id	ID	Document id
	account_number	long	Client's account number
	address	text	Client's address
	age	long	Client's age
	balance	long	Clients's account's balance
	city	text	City in which client lives
	email	text	Client's email address
	employer	text	Client's employer
	firstname	text	Client's first name
	gender	text	Client's gender
	lastname	text	Client's last name
	state	text	Two letter code of US state (e.g. Texas - TX)

### 5.1.1.2. Search index: kibana\_sample\_data\_ecommerce

#### Columns

	Name	Data type	Description / Attributes
ID	_id	ID	Document ID
[[	category	text	Product category
[[	currency	keyword	Currency of price
[[	customer_birth_date	date	Customer's birth date
[[	customer_first_name	text	Customer's first name
[[	customer_full_name	text	Customer's full name
[[	customer_gender	keyword	Customer's gender
[[	customer_id	keyword	ID of customer's document
[[	customer_last_name	text	Customer's last name
[[	customer_phone	keyword	Customer's phone
[[	day_of_week	keyword	Name of purchase weekday
[[	day_of_week_i	integer	Number of purchase weekday (Monday = 1, Tuesday = 2, . Sunday = 7)
[[	email	keyword	Customer's email address
{}	event	object	-
[[	event.dataset	keyword	-
{}	geoip	object	Geographical details provided by user
[[	geoip.city_name	keyword	City name
[[	geoip.continent_name	keyword	Continent name
[[	geoip.country_iso_code	keyword	Two letters country ISO code (e.g. Morocco - MO)
[[	geoip.location	geo_point	Coordinates of provided location
[[	geoip.region_name	keyword	Region name in country
[[	manufacturer	text	Manufacturer of purchased product
[[	order_date	date	Order date
[[	order_id	keyword	ID of order's document
{}	products	object	Purchased product details
[[	products._id	text	ID
[[	products.base_price	half_float	Base price of product
[[	products.base_unit_price	half_float	Base price of product in EUR
[[	products.category	text	Product's category
[[	products.created_on	date	Date when product was created
[[	products.discount_amount	half_float	Discount amount in base currency
[[	products.discount_percentage	half_float	Percent of base price
[[	products.manufacturer	text	Manufacturer of product

	Name	Data type	Description / Attributes
□	products.min_price	half_float	Minimal possible price of product
□	products.price	half_float	Price of product after discount
□	products.product_id	long	Product id
□	products.product_name	text	Product name
□	products.quantity	integer	Quantity of purchased product
□	products.sku	keyword	Stock keeping unit
□	products.tax_amount	half_float	Tax amount
□	products.taxful_price	half_float	Price with tax included
□	products.taxless_price	half_float	Price without tax
□	products.unit_discount_amount	half_float	Discount amount.
□	sku	keyword	Stock keeping unit
□	taxful_total_price	half_float	Total price with tax included
□	taxless_total_price	half_float	Total price without tax
□	total_quantity	integer	Total quantity
□	total_unique_products	integer	Number of unique products purchased
□	type	keyword	Type of document
□	user	keyword	Username of customer

DATA

### 5.1.1.3. Search index: kibana\_sample\_data\_flights

#### Columns

	Name	Data type	Description / Attributes
ID	_id	ID	Document ID
	AvgTicketPrice	float	Average ticket price for a flight
	Cancelled	boolean	If flight was cancelled (true/false)
	Carrier	keyword	Carrier name
	Dest	keyword	Destination airport name
	DestAirportID	keyword	Destination airport ID
	DestCityName	keyword	Destination city name
	DestCountry	keyword	Two letter code of destination country
	DestLocation	geo_point	Destination location coordinates
	DestRegion	keyword	Destination region code
	DestWeather	keyword	Weather at destination airport
	DistanceKilometers	float	Distance in kilometers
	DistanceMiles	float	Distance in miles
	FlightDelay	boolean	If flight was delayed (true / No Delay)
	FlightDelayMin	integer	Flight delay in mins
	FlightDelayType	keyword	Reason of flight delay
	FlightNum	keyword	Flight id number
	FlightTimeHour	keyword	Flight time in hours
	FlightTimeMin	float	Flight time in minutes
	Origin	keyword	Origin airport name
	OriginAirportID	keyword	Origin airport ID
	OriginCityName	keyword	Origin city name
	OriginCountry	keyword	Two letter code of origin country
	OriginLocation	geo_point	Origin location coordinates
	OriginRegion	keyword	Origin region code
	OriginWeather	keyword	Weather at origin airport
	dayOfWeek	integer	Flight take off day (Monday = 1, Tuesday = 2, ... Sunday = 7)
	timestamp	date	Document creation timestamp (datetime)

#### 5.1.1.4. Search index: kibana\_sample\_data\_logs

##### Columns

	Name	Data type	Description / Attributes
ID	_id	ID	Document ID
	@timestamp	alias	Document creation timestamp (datetime)
	agent	text	User agent that contacted server
	bytes	long	Size in bytes
	clientip	ip	IP address of client
{}	event	object	-
	event.dataset	keyword	Dataset name
	extension	text	Extension of file
{}	geo	object	-
	geo.coordinates	geo_point	GPS coordinates
	geo.dest	keyword	Destination code
	geo.src	keyword	Source code
	geo.srcdest	keyword	Code in form DestinationCode:SourceCode
	host	text	Requested host
	index	text	Index name
	ip	ip	IP address
	ip_range	ip_range	-
{}	machine	object	Machine details
	machine.os	text	Operating System on machine
	machine.ram	long	RAM installed in machine
	memory	double	Available memory
	message	text	Message sent to client
	phpmemory	long	-
	referer	keyword	Referer
	request	text	Full request address
	response	text	Reponse HTTP code
	tags	text	Tags sent with response
	timestamp	date	Timestamp of response
	timestamp_range	date_range	Possible timestamp range
	url	text	Full URL of request
	utc_time	date	UTC time of sending the response

TRIAL

## 6. Sample MongoDB Database (NoSQL Document Store)

This is documentation of sample MongoDB database - a document store (kind of NoSQL database).

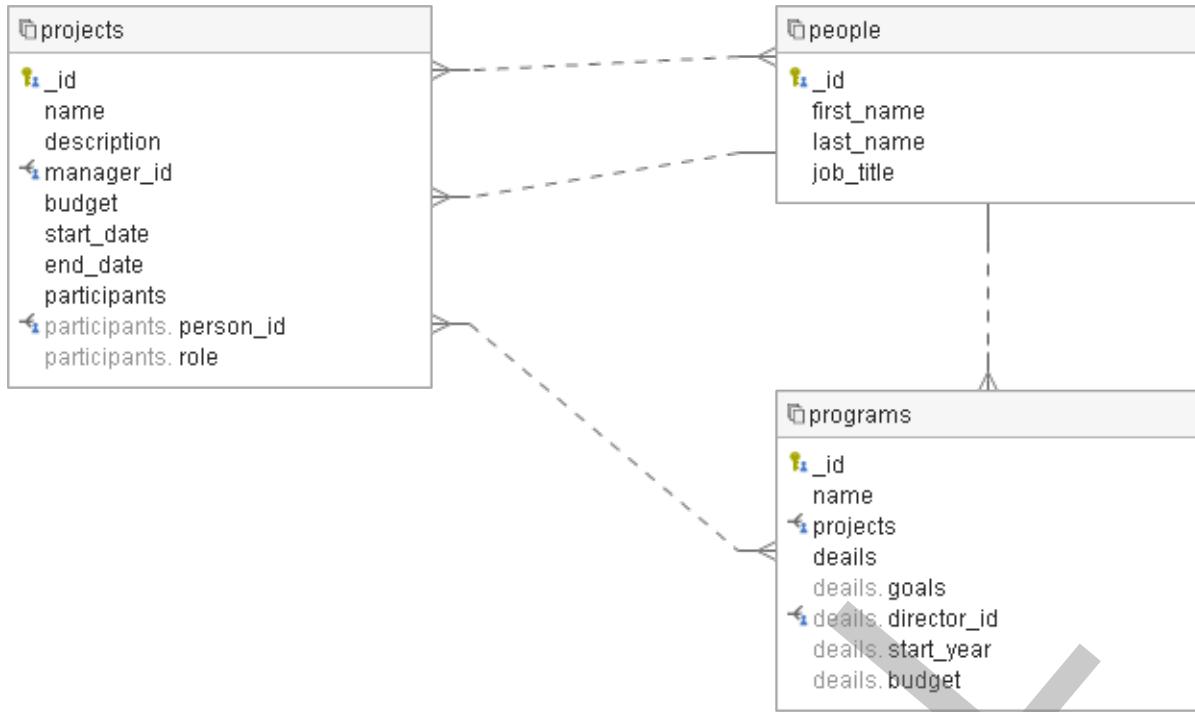
And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

## 6.1. MongoDB schema



## 6.1.1. Tables

### 6.1.1.1. Collection: people

Collection of project participants.

#### Columns

	Name	Data type	Description / Attributes
»	_id	Int32	Document ID <b>Nullable</b>
»	first_name	String	First name <b>Nullable</b>
»	last_name	String	Last name <b>Nullable</b>
»	job_title	String	Employee job title <b>Nullable</b>

#### Linked from

	Table	Join	Title / Name / Description
→	programs	<b>people_id =</b> programs.deals.director_id	fk_people_programs
↗	projects	<b>people_id =</b> projects.participants.person_id	fk_people_projects
→	projects	<b>people_id =</b> projectsmanager_id	fk_people_projects

#### Unique keys

	Columns	Name / Description
🔑	_id	pk_people_id

#### Used By

	Name
📄	people
→	programs
↗	projects
→	projects

## 6.1.1.2. Collection: programs

Collection of programs.

Columns

		Name	Data type	Description / Attributes
└─	🔑👤	_id	Int32	Document ID <b>Nullable</b>
└─	name		String	Program name <b>Nullable</b>
[ ]	projects		Int32[]	List of IDs of program projects <b>Nullable</b> References: projects
{ }	deails		Document	Program details <b>Nullable</b>
└─	deails.goals		String	A summary of program goals <b>Nullable</b>
└─	deails.director_id		Int32	Program director ID <b>Nullable</b> References: people
└─	deails.start_year		Int32	Year the program was started <b>Nullable</b>
└─	deails.budget		Int32	Program budget <b>Nullable</b>

Links to

Table		Join	Title / Name / Description
→👤	people	<b>programs</b> deails.director_id = people_id	fk_people_programs
→⌚	projects	<b>programs</b> projects = projects_id	fk_projects_programs

Unique keys

Columns		Name / Description
🔑👤	_id	pk_programs_id

Uses

Name	
📁	<b>programs</b>
→👤	people
→⌚	projects

### 6.1.1.3. Collection: projects

Collection of projects.

Columns

	Name	Data type	Description / Attributes
»	_id	Int32	Document ID <b>Nullable</b>
»	name	String	Project name <b>Nullable</b>
»	description	String	Project description <b>Nullable</b>
»	manager_id	Int32	ID of project manager <b>Nullable</b> References: people
»	budget	Int32	Project budget <b>Nullable</b>
»	start_date	String	Project start date <b>Nullable</b>
»	end_date	String	Project end date <b>Nullable</b>
[{}]	participants	Document[]	Project participants <b>Nullable</b>
»	participants.person_id	Int32	Participant person ID <b>Nullable</b> References: people
»	participants.role	String	Participant role name <b>Nullable</b>

Links to

	Table	Join	Title / Name / Description
»	people	projects.participants.person_id = people_id	fk_people_projects
»	people	projects.manager_id = people_id	fk_people_projects

Linked from

	Table	Join	Title / Name / Description
»	programs	projects_id = programs.projects	fk_projects_programs

Unique keys

	Columns	Name / Description
»	_id	pk_projects_id

Uses

	Name
»	projects
»	people
»	people

Used By

Name
 projects
 programs

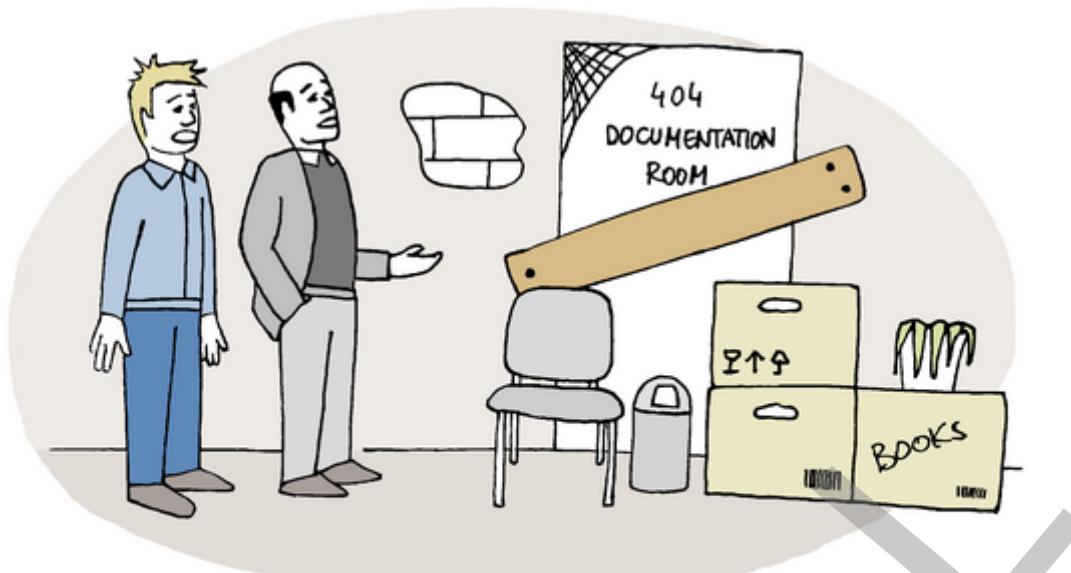
TRIAL

TRIAL

## 7. Sample Neo4j Database (NoSQL Graph Database)

This is a documentation of a sample, built-in Neo4j database "**Movies**".

And this is a random Dataedo Data Cartoon to cheer you up:



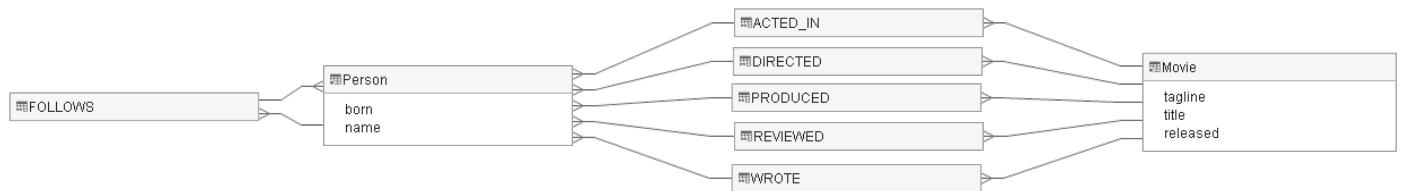
AND THIS IS A ROOM WHERE WE KEEP OUR DOCUMENTATION

Piotr@Dataedo

You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

## 7.1. Movies



TRIAL

## 7.1.1. Tables

### 7.1.1.1. Graph edge table: ACTED\_IN

Graph edge connecting Person and Movie nodes.

Columns

	Name	Data type	Description / Attributes
	roles	StringArray	Roles which an actor played in a movie.

Uses

	Name
	ACTED_IN
→	Movie

Used By

	Name
	ACTED_IN
←	Person

### 7.1.1.2. Graph edge table: DIRECTED

Graph edge connecting Person and Movie nodes.

Uses

Name
DIRECTED
→ Movie

Used By

Name
DIRECTED
← Person

TRIAL

### 7.1.1.3. Graph edge table: FOLLOW

Graph loop of Person node.

Uses

Name
 FOLLOWS
→ Person

Used By

Name
 FOLLOWS
← Person

TRIAL

#### 7.1.1.4. Graph node table: Movie

Graph node describing a movie.

Columns

Name		Data type	Description / Attributes
>tagline		String	Optional tagline - short text which serves to clarify an idea of a movie. <b>Nullable</b>
title		String	Movie's title.
released		Long	Year of a movie's release.

Used By

Name
Movie
→ ACTED_IN
→ DIRECTED
→ PRODUCED
→ REVIEWED
→ WROTE

## 7.1.1.5. Graph node table: Person

### Columns

	Name	Data type	Description / Attributes
grid icon	born	Long	The year in which a person was born. <b>Nullable</b>
grid icon	name	String	First and last name of a person.

### Uses

	Name
grid icon	Person
→	ACTED_IN
→	DIRECTED
→	FOLLOWS
→	PRODUCED
→	REVIEWED
→	WROTE

### Used By

	Name
grid icon	Person
←	FOLLOWS

### 7.1.1.6. Graph edge table: PRODUCED

Graph edge connecting Person and Movie nodes.

Uses

Name
PRODUCED
→ Movie

Used By

Name
PRODUCED
← Person

TRIAL

### 7.1.1.7. Graph edge table: REVIEWED

Graph edge connecting Person and Movie nodes.

Columns

Name		Data type	Description / Attributes
	summary	String	Summary of a review.
	rating	Long	Rating score.

Uses

Name
 REVIEWED
→ Movie

Used By

Name
 REVIEWED
← Person

TRIAL

### 7.1.1.8. Graph edge table: WROTE

Graph edge connecting Person and Movie nodes.

Uses

Name
WROTE
→ Movie

Used By

Name
WROTE
← Person

TRIAL

TRIAL

## 7.2. Other

### 7.2.1. Procedures

#### 7.2.1.1. Procedure: db.awaitIndex

Wait for an index to come online (for example: CALL db.awaitIndex("MyIndex", 300)).

##### Input/Output

	Name	Data type	Description
@	indexName	STRING?	
@	timeOutSeconds = 300	INTEGER?	
*@	Returns	VOID	

TRIAL

### 7.2.1.2. Procedure: db.awaitIndexes

Wait for all indexes to come online (for example: CALL db.awaitIndexes(300)).

#### Input/Output

	Name	Data type	Description
→@	timeOutSeconds = 300	INTEGER?	
*@	Returns	VOID	

TRIAL

### 7.2.1.3. Procedure: db.checkpoint

Initiate and wait for a new check point, or wait any already on-going check point to complete. Note that this temporarily disables the `dbms.checkpoint.iops.limit` setting in order to make the check point complete faster. This might cause transaction throughput to degrade slightly, due to increased IO load.

#### Input/Output

	Name	Data type	Description
*@	success	BOOLEAN?	
*@	message	STRING?	

TRIAL

#### 7.2.1.4. Procedure: db.clearQueryCaches

Clears all query caches.

Input/Output

Name	Data type	Description
*@ value	STRING?	

TRIAL

### 7.2.1.5. Procedure: db.constraints

List all constraints in the database.

Input/Output

	Name	Data type	Description
*@	name	STRING?	
*@	description	STRING?	
*@	details	STRING?	

TRIAL

### 7.2.1.6. Procedure: db.createIndex

Create a named schema index with specified index provider and configuration (optional). Yield: name, labels, properties, providerName, status

#### Input/Output

	Name	Data type	Description
→@	indexName	STRING?	
→@	labels	LIST? OF STRING?	
→@	properties	LIST? OF STRING?	
→@	providerName	STRING?	
→@	config = {}	MAP?	
*@	name	STRING?	
*@	labels	LIST? OF STRING?	
*@	properties	LIST? OF STRING?	
*@	providerName	STRING?	
*@	status	STRING?	

TRIAL

### 7.2.1.7. Procedure: db.createLabel

#### Create a label

##### Input/Output

	Name	Data type	Description
@	newLabel	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.8. Procedure: db.createNodeKey

Create a named node key constraint. Backing index will use specified index provider and configuration (optional).  
Yield: name, labels, properties, providerName, status

#### Input/Output

	Name	Data type	Description
@	constraintName	STRING?	
@	labels	LIST? OF STRING?	
@	properties	LIST? OF STRING?	
@	providerName	STRING?	
@	config = {}	MAP?	
@	name	STRING?	
@	labels	LIST? OF STRING?	
@	properties	LIST? OF STRING?	
@	providerName	STRING?	
@	status	STRING?	

TRIAL

### 7.2.1.9. Procedure: db.createProperty

#### Create a Property

##### Input/Output

	Name	Data type	Description
@>	newProperty	STRING?	
*@>	Returns	VOID	

TRIAL

### 7.2.1.10. Procedure: db.createRelationshipType

Create a RelationshipType

Input/Output

	Name	Data type	Description
→@	newRelationshipType	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.11. Procedure: db.createUniquePropertyConstraint

Create a named unique property constraint. Backing index will use specified index provider and configuration (optional). Yield: name, labels, properties, providerName, status

#### Input/Output

	Name	Data type	Description
*@	constraintName	STRING?	
*@	labels	LIST? OF STRING?	
*@	properties	LIST? OF STRING?	
*@	providerName	STRING?	
*@	config = {}	MAP?	
*@	name	STRING?	
*@	labels	LIST? OF STRING?	
*@	properties	LIST? OF STRING?	
*@	providerName	STRING?	
*@	status	STRING?	

TRIAL

### 7.2.1.12. Procedure: db.index.fulltext.awaitEventuallyConsistentIndexRefresh

Wait for the updates from recently committed transactions to be applied to any eventually-consistent full-text indexes.

#### Input/Output

Name	Data type	Description
*@ Returns	VOID	

TRIAL

### 7.2.1.13. Procedure: db.index.fulltext.createNodeIndex

Create a node full-text index for the given labels and properties. The optional 'config' map parameter can be used to supply settings to the index. Supported settings are 'analyzer', for specifying what analyzer to use when indexing and querying. Use the `db.index.fulltext.listAvailableAnalyzers` procedure to see what options are available. And 'eventually\_consistent' which can be set to 'true' to make this index eventually consistent, such that updates from committing transactions are applied in a background thread.

#### Input/Output

	Name	Data type	Description
@	indexName	STRING?	
@	labels	LIST? OF STRING?	
@	properties	LIST? OF STRING?	
@	config = {}	MAP?	
*	Returns	VOID	

TRIAL

#### 7.2.1.14. Procedure: db.index.fulltext.createRelationshipIndex

Create a relationship full-text index for the given relationship types and properties. The optional 'config' map parameter can be used to supply settings to the index. Supported settings are 'analyzer', for specifying what analyzer to use when indexing and querying. Use the `db.index.fulltext.listAvailableAnalyzers` procedure to see what options are available. And 'eventually\_consistent' which can be set to 'true' to make this index eventually consistent, such that updates from committing transactions are applied in a background thread.

##### Input/Output

	Name	Data type	Description
@	indexName	STRING?	
@	relationshipTypes	LIST? OF STRING?	
@	properties	LIST? OF STRING?	
@	config = {}	MAP?	
*	Returns	VOID	

TRIAL

### 7.2.1.15. Procedure: db.index.fulltext.drop

Drop the specified index.

Input/Output

	Name	Data type	Description
→@	indexName	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.16. Procedure: db.index.fulltext.listAvailableAnalyzers

List the available analyzers that the full-text indexes can be configured with.

#### Input/Output

	Name	Data type	Description
*@>	analyzer	STRING?	
*@>	description	STRING?	
*@>	stopwords	LIST? OF STRING?	

TRIAL

### 7.2.1.17. Procedure: db.index.fulltext.queryNodes

Query the given full-text index. Returns the matching nodes, and their Lucene query score, ordered by score. Valid keys for the options map are: 'skip' to skip the top N results; 'limit' to limit the number of results returned.

#### Input/Output

	Name	Data type	Description
>@	indexName	STRING?	
>@	queryString	STRING?	
>@	options = {}	MAP?	
*@>	node	NODE?	
*@>	score	FLOAT?	

TRIAL

### 7.2.1.18. Procedure: db.index.fulltext.queryRelationships

Query the given full-text index. Returns the matching relationships, and their Lucene query score, ordered by score. Valid keys for the options map are: 'skip' to skip the top N results; 'limit' to limit the number of results returned.

#### Input/Output

	Name	Data type	Description
>@	indexName	STRING?	
>@	queryString	STRING?	
>@	options = {}	MAP?	
*@>	relationship	RELATIONSHIP?	
*@>	score	FLOAT?	

TRIAL

### 7.2.1.19. Procedure: db.indexDetails

Detailed description of specific index.

Input/Output

	Name	Data type	Description
@	indexName	STRING?	
@	id	INTEGER?	
@	name	STRING?	
@	state	STRING?	
@	populationPercent	FLOAT?	
@	uniqueness	STRING?	
@	type	STRING?	
@	entityType	STRING?	
@	labelsOrTypes	LIST? OF STRING?	
@	properties	LIST? OF STRING?	
@	provider	STRING?	
@	indexConfig	MAP?	
@	failureMessage	STRING?	

### 7.2.1.20. Procedure: db.indexes

List all indexes in the database.

Input/Output

	Name	Data type	Description
*@	id	INTEGER?	
*@	name	STRING?	
*@	state	STRING?	
*@	populationPercent	FLOAT?	
*@	uniqueness	STRING?	
*@	type	STRING?	
*@	entityType	STRING?	
*@	labelsOrTypes	LIST? OF STRING?	
*@	properties	LIST? OF STRING?	
*@	provider	STRING?	

TRIAL

### 7.2.1.21. Procedure: db.info

Provides information regarding the database.

#### Input/Output

	Name	Data type	Description
*@	id	STRING?	
*@	name	STRING?	
*@	creationDate	STRING?	

TRIAL

### 7.2.1.22. Procedure: db.labels

List all available labels in the database.

Input/Output

	Name	Data type	Description
*@>	label	STRING?	

TRIAL

### 7.2.1.23. Procedure: db.listLocks

List all locks at this database.

Input/Output

	Name	Data type	Description
*@	mode	STRING?	
*@	resourceType	STRING?	
*@	resourceId	INTEGER?	
*@	transactionId	STRING?	

TRIAL

### 7.2.1.24. Procedure: db.ping

This procedure can be used by client side tooling to test whether they are correctly connected to a database. The procedure is available in all databases and always returns true. A faulty connection can be detected by not being able to call this procedure.

#### Input/Output

Name	Data type	Description
*@ success	BOOLEAN?	

TRIAL

### 7.2.1.25. Procedure: db.prepareForReplanning

Triggers an index resample and waits for it to complete, and after that clears query caches. After this procedure has finished queries will be planned using the latest database statistics.

#### Input/Output

	Name	Data type	Description
→@	timeOutSeconds = 300	INTEGER?	
*@	Returns	VOID	

TRIAL

### 7.2.1.26. Procedure: db.propertyKeys

List all property keys in the database.

Input/Output

	Name	Data type	Description
*@	propertyKey	STRING?	

TRIAL

### 7.2.1.27. Procedure: db.relationshipTypes

List all available relationship types in the database.

Input/Output

	Name	Data type	Description
*@	relationshipType	STRING?	

TRIAL

### 7.2.1.28. Procedure: db.resampleIndex

Schedule resampling of an index (for example: CALL db.resampleIndex("MyIndex")).

#### Input/Output

	Name	Data type	Description
@	indexName	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.29. Procedure: db.resampleOutdatedIndexes

Schedule resampling of all outdated indexes.

#### Input/Output

	Name	Data type	Description
*@	Returns	VOID	

TRIAL

### 7.2.1.30. Procedure: db.schema.nodeTypeProperties

Show the derived property schema of the nodes in tabular form.

Input/Output

	Name	Data type	Description
*@	nodeType	STRING?	
*@	nodeLabels	LIST? OF STRING?	
*@	propertyName	STRING?	
*@	propertyTypes	LIST? OF STRING?	
*@	mandatory	BOOLEAN?	

TRIAL

### 7.2.1.31. Procedure: db.schema.relTypeProperties

Show the derived property schema of the relationships in tabular form.

Input/Output

	Name	Data type	Description
*@	relType	STRING?	
*@	propertyName	STRING?	
*@	propertyTypes	LIST? OF STRING?	
*@	mandatory	BOOLEAN?	

TRIAL

### 7.2.1.32. Procedure: db.schema.visualization

Visualize the schema of the data.

Input/Output

	Name	Data type	Description
*@	nodes	LIST? OF NODE?	
*@	relationships	LIST? OF RELATIONSHIP?	

TRIAL

### 7.2.1.33. Procedure: db.schemaStatements

List all statements for creating and dropping existing indexes and constraints. Note that only index types introduced before Neo4j 4.3 are included.

#### Input/Output

	Name	Data type	Description
*@	name	STRING?	
*@	type	STRING?	
*@	createStatement	STRING?	
*@	dropStatement	STRING?	

TRIAL

#### 7.2.1.34. Procedure: db.stats.clear

Clear collected data of a given data section. Valid sections are 'QUERIES'

##### Input/Output

	Name	Data type	Description
@	section	STRING?	
*@	section	STRING?	
@	success	BOOLEAN?	
*@	message	STRING?	

TRIAL

### 7.2.1.35. Procedure: db.stats.collect

Start data collection of a given data section. Valid sections are 'QUERIES'

#### Input/Output

	Name	Data type	Description
@	section	STRING?	
@	config = {}	MAP?	
@	section	STRING?	
@	success	BOOLEAN?	
@	message	STRING?	

TRIAL

### 7.2.1.36. Procedure: db.stats.retrieve

Retrieve statistical data about the current database. Valid sections are 'GRAPH COUNTS', 'TOKENS', 'QUERIES', 'META'

#### Input/Output

	Name	Data type	Description
@	section	STRING?	
@	config = {}	MAP?	
@	section	STRING?	
*	data	MAP?	

TRIAL

### 7.2.1.37. Procedure: db.stats.retrieveAllAnonymized

Retrieve all available statistical data about the current database, in an anonymized form.

#### Input/Output

	Name	Data type	Description
@	graphToken	STRING?	
@	config = {}	MAP?	
@	section	STRING?	
*	data	MAP?	

TRIAL

### 7.2.1.38. Procedure: db.stats.status

Retrieve the status of all available collector daemons, for this database.

#### Input/Output

	Name	Data type	Description
*@>	section	STRING?	
*@>	status	STRING?	
*@>	data	MAP?	

TRIAL

### 7.2.1.39. Procedure: db.stats.stop

Stop data collection of a given data section. Valid sections are 'QUERIES'

#### Input/Output

	Name	Data type	Description
@	section	STRING?	
*@	section	STRING?	
*@	success	BOOLEAN?	
*@	message	STRING?	

TRIAL

#### 7.2.1.40. Procedure: dbms.cluster.overview

Overview of all currently accessible cluster members, their databases and roles.

##### Input/Output

	Name	Data type	Description
*@	id	STRING?	
*@	addresses	LIST? OF STRING?	
*@	databases	MAP?	
*@	groups	LIST? OF STRING?	

TRIAL

### 7.2.1.41. Procedure: dbms.cluster.protocols

#### Overview of installed protocols

##### Input/Output

	Name	Data type	Description
*@	orientation	STRING?	
*@	remoteAddress	STRING?	
*@	applicationProtocol	STRING?	
*@	applicationProtocolVersion	INTEGER?	
*@	modifierProtocols	STRING?	

TRIAL

#### 7.2.1.42. Procedure: dbms.cluster.role

The role of this instance in the cluster for the specified database.

Input/Output

	Name	Data type	Description
→@	database	STRING?	
*@	role	STRING?	

TRIAL

### 7.2.1.43. Procedure: dbms.cluster.routing.getRoutingTable

Returns endpoints of this instance.

#### Input/Output

	Name	Data type	Description
@	context	MAP?	
@	database = null	STRING?	
@	ttl	INTEGER?	
*	servers	LIST? OF MAP?	

TRIAL

#### 7.2.1.44. Procedure: dbms.components

List DBMS components and their versions.

Input/Output

	Name	Data type	Description
*@	name	STRING?	
*@	versions	LIST? OF STRING?	
*@	edition	STRING?	

TRIAL

#### 7.2.1.45. Procedure: dbms.database.state

The actual status of the database with the provided name on this neo4j instance.

##### Input/Output

	Name	Data type	Description
@	databaseName	STRING?	
*@	role	STRING?	
*@	address	STRING?	
*@	status	STRING?	
*@	error	STRING?	

TRIAL

### 7.2.1.46. Procedure: dbms.functions

List all functions in the DBMS.

Input/Output

	Name	Data type	Description
*@	name	STRING?	
*@	signature	STRING?	
*@	category	STRING?	
*@	description	STRING?	
*@	aggregating	BOOLEAN?	
*@	defaultBuiltInRoles	LIST? OF STRING?	

TRIAL

#### 7.2.1.47. Procedure: dbms.info

Provides information regarding the DBMS.

Input/Output

	Name	Data type	Description
*@	id	STRING?	
*@	name	STRING?	
*@	creationDate	STRING?	

TRIAL

#### 7.2.1.48. Procedure: dbms.killConnection

Kill network connection with the given connection id.

Input/Output

	Name	Data type	Description
@	id	STRING?	
*@	connectionId	STRING?	
*@	username	STRING?	
*@	message	STRING?	

TRIAL

#### 7.2.1.49. Procedure: dbms.killConnections

Kill all network connections with the given connection ids.

##### Input/Output

	Name	Data type	Description
→@	ids	LIST? OF STRING?	
*@	connectionId	STRING?	
*@	username	STRING?	
*@	message	STRING?	

TRIAL

### 7.2.1.50. Procedure: dbms.killQueries

Kill all transactions executing a query with any of the given query ids.

Input/Output

	Name	Data type	Description
→@	ids	LIST? OF STRING?	
*@	queryId	STRING?	
*@	username	STRING?	
*@	message	STRING?	

TRIAL

### 7.2.1.51. Procedure: dbms.killQuery

Kill all transactions executing the query with the given query id.

#### Input/Output

	Name	Data type	Description
→@	id	STRING?	
*@	queryId	STRING?	
*@	username	STRING?	
*@	message	STRING?	

TRIAL

### 7.2.1.52. Procedure: dbms.killTransaction

Kill transaction with provided id.

Input/Output

	Name	Data type	Description
@	id	STRING?	
*@	transactionId	STRING?	
*@	username	STRING?	
*@	message	STRING?	

TRIAL

### 7.2.1.53. Procedure: dbms.killTransactions

Kill transactions with provided ids.

Input/Output

	Name	Data type	Description
@	ids	LIST? OF STRING?	
*@	transactionId	STRING?	
*@	username	STRING?	
*@	message	STRING?	

TRIAL

#### 7.2.1.54. Procedure: dbms.listActiveLocks

List the active lock requests granted for the transaction executing the query with the given query id.

##### Input/Output

	Name	Data type	Description
@	queryId	STRING?	
*@	mode	STRING?	
*@	resourceType	STRING?	
*@	resourceId	INTEGER?	

TRIAL

### 7.2.1.55. Procedure: dbms.listCapabilities

#### List capabilities

##### Input/Output

	Name	Data type	Description
*@	name	STRING?	
*@	description	STRING?	
*@	value	ANY?	

TRIAL

### 7.2.1.56. Procedure: dbms.listConfig

List the currently active config of Neo4j.

Input/Output

	Name	Data type	Description
@	searchString =	STRING?	
*	name	STRING?	
*	description	STRING?	
*	value	STRING?	
*	dynamic	BOOLEAN?	

TRIAL

### 7.2.1.57. Procedure: dbms.listConnections

List all accepted network connections at this instance that are visible to the user.

#### Input/Output

	Name	Data type	Description
*@	connectionId	STRING?	
*@	connectTime	STRING?	
*@	connector	STRING?	
*@	username	STRING?	
*@	userAgent	STRING?	
*@	serverAddress	STRING?	
*@	clientAddress	STRING?	

TRIAL

### 7.2.1.58. Procedure: dbms.listPools

List all memory pools, including sub pools, currently registered at this instance that are visible to the user.

#### Input/Output

	Name	Data type	Description
*@	pool	STRING?	
*@	databaseName	STRING?	
*@	heapMemoryUsed	STRING?	
*@	heapMemoryUsedBytes	STRING?	
*@	nativeMemoryUsed	STRING?	
*@	nativeMemoryUsedBytes	STRING?	
*@	freeMemory	STRING?	
*@	freeMemoryBytes	STRING?	
*@	totalPoolMemory	STRING?	
*@	totalPoolMemoryBytes	STRING?	

TRIAL

### 7.2.1.59. Procedure: dbms.listQueries

List all queries currently executing at this instance that are visible to the user.

#### Input/Output

	Name	Data type	Description
*@	queryId	STRING?	
*@	username	STRING?	
*@	metaData	MAP?	
*@	query	STRING?	
*@	parameters	MAP?	
*@	planner	STRING?	
*@	runtime	STRING?	
*@	indexes	LIST? OF MAP?	
*@	startTime	STRING?	
*@	protocol	STRING?	
*@	clientAddress	STRING?	
*@	requestUri	STRING?	
*@	status	STRING?	
*@	resourceInformation	MAP?	
*@	activeLockCount	INTEGER?	
*@	elapsedTimeMillis	INTEGER?	
*@	cpuTimeMillis	INTEGER?	
*@	waitTimeMillis	INTEGER?	
*@	idleTimeMillis	INTEGER?	
*@	allocatedBytes	INTEGER?	
*@	pageHits	INTEGER?	
*@	pageFaults	INTEGER?	
*@	connectionId	STRING?	
*@	database	STRING?	

### 7.2.1.60. Procedure: dbms.listTransactions

List all transactions currently executing at this instance that are visible to the user.

#### Input/Output

	Name	Data type	Description
*@	transactionId	STRING?	
*@	username	STRING?	
*@	metaData	MAP?	
*@	startTime	STRING?	
*@	protocol	STRING?	
*@	clientAddress	STRING?	
*@	requestUri	STRING?	
*@	currentQueryId	STRING?	
*@	currentQuery	STRING?	
*@	activeLockCount	INTEGER?	
*@	status	STRING?	
*@	resourceInformation	MAP?	
*@	elapsedTimeMillis	INTEGER?	
*@	cpuTimeMillis	INTEGER?	
*@	waitTimeMillis	INTEGER?	
*@	idleTimeMillis	INTEGER?	
*@	allocatedBytes	INTEGER?	
*@	allocatedDirectBytes	INTEGER?	
*@	pageHits	INTEGER?	
*@	pageFaults	INTEGER?	
*@	connectionId	STRING?	
*@	initializationStackTrace	STRING?	
*@	database	STRING?	
*@	estimatedUsedHeapMemory	INTEGER?	

### 7.2.1.61. Procedure: dbms.procedures

List all procedures in the DBMS.

Input/Output

	Name	Data type	Description
*@	name	STRING?	
*@	signature	STRING?	
*@	description	STRING?	
*@	mode	STRING?	
*@	defaultBuiltInRoles	LIST? OF STRING?	
*@	worksOnSystem	BOOLEAN?	

TRIAL

### 7.2.1.62. Procedure: dbms.quarantineDatabase

Place a database into quarantine or remove from it.

#### Input/Output

	Name	Data type	Description
@	databaseName	STRING?	
@	setStatus	BOOLEAN?	
@	reason = No reason given	STRING?	
*	databaseName	STRING?	
*	quarantined	BOOLEAN?	
*	result	STRING?	

TRIAL

### 7.2.1.63. Procedure: dbms.queryJmx

Query JMX management data by domain and name. For instance, "\*:\*"

#### Input/Output

	Name	Data type	Description
@	query	STRING?	
*@	name	STRING?	
@	description	STRING?	
*@	attributes	MAP?	

TRIAL

#### 7.2.1.64. Procedure: dbms.routing.getRoutingTable

Returns endpoints of this instance.

##### Input/Output

	Name	Data type	Description
@	context	MAP?	
@	database = null	STRING?	
@	ttl	INTEGER?	
*	servers	LIST? OF MAP?	

TRIAL

### 7.2.1.65. Procedure: dbms.scheduler.failedJobs

List failed job runs. There is a limit for amount of historical data.

#### Input/Output

	Name	Data type	Description
*@	jobId	STRING?	
*@	group	STRING?	
*@	database	STRING?	
*@	submitter	STRING?	
*@	description	STRING?	
*@	type	STRING?	
*@	submitted	STRING?	
*@	executionStart	STRING?	
*@	failureTime	STRING?	
*@	failureDescription	STRING?	

TRIAL

### 7.2.1.66. Procedure: dbms.scheduler.groups

List the job groups that are active in the database internal job scheduler.

Input/Output

	Name	Data type	Description
*@	group	STRING?	
*@	threads	INTEGER?	

TRIAL

### 7.2.1.67. Procedure: dbms.scheduler.jobs

List all jobs that are active in the database internal job scheduler.

#### Input/Output

	Name	Data type	Description
*@	jobId	STRING?	
*@	group	STRING?	
*@	submitted	STRING?	
*@	database	STRING?	
*@	submitter	STRING?	
*@	description	STRING?	
*@	type	STRING?	
*@	scheduledAt	STRING?	
*@	period	STRING?	
*@	state	STRING?	
*@	currentStateDescription	STRING?	

### 7.2.1.68. Procedure: dbms.scheduler.profile

Begin profiling all threads within the given job group, for the specified duration. Note that profiling incurs overhead to a system, and will slow it down.

#### Input/Output

	Name	Data type	Description
→@	method	STRING?	
→@	group	STRING?	
→@	duration	STRING?	
*@*	profile	STRING?	

TRIAL

### 7.2.1.69. Procedure: dbms.security.activateUser

Activate a suspended user.

Input/Output

	Name	Data type	Description
@	username	STRING?	
@	requirePasswordChange = true	BOOLEAN?	
*@	Returns	VOID	

TRIAL

### 7.2.1.70. Procedure: dbms.security.addRoleToUser

Assign a role to the user.

Input/Output

	Name	Data type	Description
@	roleName	STRING?	
@	username	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.71. Procedure: dbms.security.changePassword

Change the current user's password.

Input/Output

	Name	Data type	Description
@	password	STRING?	
@	requirePasswordChange = false	BOOLEAN?	
*@	Returns	VOID	

TRIAL

### 7.2.1.72. Procedure: dbms.security.changeUserPassword

Change the given user's password.

Input/Output

	Name	Data type	Description
@	username	STRING?	
@	newPassword	STRING?	
@	requirePasswordChange = true	BOOLEAN?	
*	Returns	VOID	

TRIAL

### 7.2.1.73. Procedure: dbms.security.clearAuthCache

Clears authentication and authorization cache.

#### Input/Output

	Name	Data type	Description
*@	Returns	VOID	

TRIAL

#### 7.2.1.74. Procedure: dbms.security.createRole

Create a new role.

Input/Output

	Name	Data type	Description
→@	roleName	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.75. Procedure: dbms.security.createUser

Create a new user.

Input/Output

	Name	Data type	Description
@	username	STRING?	
@	password	STRING?	
@	requirePasswordChange = true	BOOLEAN?	
*	Returns	VOID	

TRIAL

### 7.2.1.76. Procedure: dbms.security.deleteRole

Delete the specified role. Any role assignments will be removed.

#### Input/Output

	Name	Data type	Description
→@	roleName	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.77. Procedure: dbms.security.deleteUser

Delete the specified user.

Input/Output

	Name	Data type	Description
→@	username	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.78. Procedure: dbms.security.listRoles

List all available roles.

Input/Output

	Name	Data type	Description
*@	role	STRING?	
*@	users	LIST? OF STRING?	

TRIAL

### 7.2.1.79. Procedure: dbms.security.listRolesForUser

List all roles assigned to the specified user.

Input/Output

	Name	Data type	Description
→@	username	STRING?	
*@	value	STRING?	

TRIAL

### 7.2.1.80. Procedure: dbms.security.listUsers

List all native users.

Input/Output

	Name	Data type	Description
*@	username	STRING?	
*@	roles	LIST? OF STRING?	
*@	flags	LIST? OF STRING?	

TRIAL

### 7.2.1.81. Procedure: dbms.security.listUsersForRole

List all users currently assigned the specified role.

Input/Output

	Name	Data type	Description
→@	roleName	STRING?	
*@	value	STRING?	

TRIAL

### 7.2.1.82. Procedure: dbms.security.removeRoleFromUser

Unassign a role from the user.

Input/Output

	Name	Data type	Description
@	roleName	STRING?	
@	username	STRING?	
*@	Returns	VOID	

TRIAL

### 7.2.1.83. Procedure: dbms.security.suspendUser

Suspend the specified user.

#### Input/Output

	Name	Data type	Description
→@	username	STRING?	
*@	Returns	VOID	

TRIAL

#### 7.2.1.84. Procedure: dbms.setConfigValue

Updates a given setting value. Passing an empty value will result in removing the configured value and falling back to the default value. Changes will not persist and will be lost if the server is restarted.

##### Input/Output

	Name	Data type	Description
@	setting	STRING?	
@	value	STRING?	
*	Returns	VOID	

TRIAL

### 7.2.1.85. Procedure: dbms.showCurrentUser

Show the current user.

Input/Output

	Name	Data type	Description
*@	username	STRING?	
*@	roles	LIST? OF STRING?	
*@	flags	LIST? OF STRING?	

TRIAL

### 7.2.1.86. Procedure: dbms.upgrade

Upgrade the system database schema if it is not the current schema.

#### Input/Output

	Name	Data type	Description
*@	status	STRING?	
*@	upgradeResult	STRING?	

TRIAL

### 7.2.1.87. Procedure: dbms.upgradeStatus

Report the current status of the system database sub-graph schema.

#### Input/Output

	Name	Data type	Description
*@	status	STRING?	
*@	description	STRING?	
*@	resolution	STRING?	

TRIAL

### 7.2.1.88. Procedure: jwt.security.requestAccess

#### Input/Output

	Name	Data type	Description
→@	application	STRING?	
*@	token	STRING?	

TRIAL

### 7.2.1.89. Procedure: tx.getMetaData

Provides attached transaction metadata.

Input/Output

	Name	Data type	Description
*@	metadata	MAP?	

TRIAL

### 7.2.1.90. Procedure: tx.setMetaData

Attaches a map of data to the transaction. The data will be printed when listing queries, and inserted into the query log.

#### Input/Output

	Name	Data type	Description
→@	data	MAP?	
*@	Returns	VOID	

TRIAL

## 7.2.2. Functions

### 7.2.2.1. Function: abs

Returns the absolute value of an integer.

Input/Output

	Name	Data type	Description
@	input	INTEGER?	
@	input	FLOAT?	
@	input	INTEGER?	
@	input	FLOAT?	
*	Returns	INTEGER?	
*	Returns	FLOAT?	
*	Returns	INTEGER?	
*	Returns	FLOAT?	

TRIAL

### 7.2.2.2. Function: abs

Returns the absolute value of a floating point number.

TRIAL

### 7.2.2.3. Function: acos

Returns the arccosine of a number in radians.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

#### 7.2.2.4. Function: all

Returns true if the predicate holds for all elements in the given list.

Input/Output

Name	Data type	Description
*@ Returns	BOOLEAN?	

TRIAL

### 7.2.2.5. Function: any

Returns true if the predicate holds for at least one element in the given list.

Input/Output

Name	Data type	Description
*@ Returns	BOOLEAN?	

TRIAL

### 7.2.2.6. Function: asin

Returns the arcsine of a number in radians.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.7. Function: atan

Returns the arctangent of a number in radians.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.8. Function: atan2

Returns the arctangent2 of a set of coordinates in radians.

Input/Output

	Name	Data type	Description
@	y	FLOAT?	
@	x	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.9. Function: avg

Returns the average of a set of integer values.

#### Input/Output

	Name	Data type	Description
@	input	INTEGER?	
@	input	FLOAT?	
@	input	DURATION?	
@	input	INTEGER?	
@	input	FLOAT?	
@	input	DURATION?	
@	input	INTEGER?	
@	input	FLOAT?	
@	input	DURATION?	
*	Returns	INTEGER?	
*	Returns	FLOAT?	
*	Returns	DURATION?	
*	Returns	INTEGER?	
*	Returns	FLOAT?	
*	Returns	DURATION?	
*	Returns	INTEGER?	
*	Returns	FLOAT?	
*	Returns	DURATION?	

#### 7.2.2.10. Function: avg

Returns the average of a set of floating point values.

TRIAL

### 7.2.2.11. Function: avg

Returns the average of a set of duration values.

TRIAL

### 7.2.2.12. Function: ceil

Returns the smallest floating point number that is greater than or equal to a number and equal to a mathematical integer.

#### Input/Output

	Name	Data type	Description
→@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.13. Function: coalesce

Returns the first non-null value in a list of expressions.

#### Input/Output

	Name	Data type	Description
@	input	ANY?	
*@	Returns	ANY?	

TRIAL

#### 7.2.2.14. Function: collect

Returns a list containing the values returned by an expression.

##### Input/Output

	Name	Data type	Description
>@	input	ANY?	
*@	Returns	LIST? OF ANY?	

TRIAL

### 7.2.2.15. Function: cos

Returns the cosine of a number.

Input/Output

	Name	Data type	Description
→@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.16. Function: cot

Returns the cotangent of a number.

Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.17. Function: count

Returns the number of values or rows.

#### Input/Output

	Name	Data type	Description
@	input	ANY?	
*@	Returns	INTEGER?	

TRIAL

### 7.2.2.18. Function: date

Create a Date instant.

Input/Output

	Name	Data type	Description
→@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATE?	

TRIAL

### 7.2.2.19. Function: date.realtime

Get the current Date instant using the realtime clock.

Input/Output

	Name	Data type	Description
@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATE?	

TRIAL

### 7.2.2.20. Function: date.statement

Get the current Date instant using the statement clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATE?	

TRIAL

### 7.2.2.21. Function: date.transaction

Get the current Date instant using the transaction clock.

Input/Output

	Name	Data type	Description
@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATE?	

TRIAL

### 7.2.2.22. Function: date.truncate

Truncate the input temporal value to a Date instant using the specified unit.

#### Input/Output

	Name	Data type	Description
@	unit	STRING?	
@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
@	fields = null	MAP?	
*	Returns	DATE?	

TRIAL

### 7.2.2.23. Function: datetime

Create a DateTime instant.

Input/Output

	Name	Data type	Description
→@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATETIME?	

TRIAL

#### 7.2.2.24. Function: datetime.fromepoch

Create a DateTime given the seconds and nanoseconds since the start of the epoch.

Input/Output

	Name	Data type	Description
@	seconds	NUMBER?	
@	nanoseconds	NUMBER?	
*@	Returns	DATETIME?	

TRIAL

### 7.2.2.25. Function: `datetime.fromepochmillis`

Create a DateTime given the milliseconds since the start of the epoch.

Input/Output

	Name	Data type	Description
→@	milliseconds	NUMBER?	
*@	Returns	DATETIME?	

TRIAL

### 7.2.2.26. Function: `datetime.realtime`

Get the current DateTime instant using the realtime clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATETIME?	

TRIAL

### 7.2.2.27. Function: datetime.statement

Get the current DateTime instant using the statement clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATETIME?	

TRIAL

### 7.2.2.28. Function: datetime.transaction

Get the current DateTime instant using the transaction clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	DATETIME?	

TRIAL

### 7.2.2.29. Function: datetime.truncate

Truncate the input temporal value to a DateTime instant using the specified unit.

Input/Output

	Name	Data type	Description
@	unit	STRING?	
@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
@	fields = null	MAP?	
*	Returns	DATETIME?	

TRIAL

### 7.2.2.30. Function: degrees

Converts radians to degrees.

Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.31. Function: distance

Returns a floating point number representing the geodesic distance between any two points in the same CRS.

#### Input/Output

	Name	Data type	Description
@from	from	POINT?	
@to	to	POINT?	
*@Returns	Returns	FLOAT?	

TRIAL

### 7.2.2.32. Function: duration

Construct a Duration value.

Input/Output

	Name	Data type	Description
→@	input	ANY?	
*@	Returns	DURATION?	

TRIAL

### 7.2.2.33. Function: duration.between

Compute the duration between the 'from' instant (inclusive) and the 'to' instant (exclusive) in logical units.

#### Input/Output

	Name	Data type	Description
@from	from	ANY?	
@to	to	ANY?	
*@Returns	Returns	DURATION?	

TRIAL

#### 7.2.2.34. Function: duration.inDays

Compute the duration between the 'from' instant (inclusive) and the 'to' instant (exclusive) in days.

Input/Output

	Name	Data type	Description
@	from	ANY?	
@	to	ANY?	
*@	Returns	DURATION?	

TRIAL

### 7.2.2.35. Function: duration.inMonths

Compute the duration between the 'from' instant (inclusive) and the 'to' instant (exclusive) in months.

Input/Output

	Name	Data type	Description
@	from	ANY?	
@	to	ANY?	
*@	Returns	DURATION?	

TRIAL

### 7.2.2.36. Function: duration.inSeconds

Compute the duration between the 'from' instant (inclusive) and the 'to' instant (exclusive) in seconds.

#### Input/Output

	Name	Data type	Description
@	from	ANY?	
@	to	ANY?	
*@	Returns	DURATION?	

TRIAL

### 7.2.2.37. Function: e

Returns the base of the natural logarithm, e.

Input/Output

	Name	Data type	Description
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.38. Function: endNode

Returns the end node of a relationship.

Input/Output

	Name	Data type	Description
→@	input	RELATIONSHIP?	
*@	Returns	NODE?	

TRIAL

### 7.2.2.39. Function: exists

Returns true if a match for the pattern exists in the graph, or if the specified property exists in the node, relationship or map.

#### Input/Output

	Name	Data type	Description
→@	input	ANY?	
*@	Returns	BOOLEAN?	

TRIAL

#### 7.2.2.40. Function: exp

Returns  $e^n$ , where  $e$  is the base of the natural logarithm, and  $n$  is the value of the argument expression.

##### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

#### 7.2.2.41. Function: file

Returns the absolute path of the file that LOAD CSV is using.

Input/Output

	Name	Data type	Description
*@	Returns	STRING?	

TRIAL

#### 7.2.2.42. Function: floor

Returns the largest floating point number that is less than or equal to a number and equal to a mathematical integer.

##### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

#### 7.2.2.43. Function: haversin

Returns half the versine of a number.

Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

#### 7.2.2.44. Function: head

Returns the first element in a list.

Input/Output

	Name	Data type	Description
→@	list	LIST? OF ANY?	
*@	Returns	ANY?	

TRIAL

#### 7.2.2.45. Function: id

Returns the id of a node.

Input/Output

	Name	Data type	Description
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	NODE?	
@@	input	RELATIONSHIP?	
*@	Returns	INTEGER?	

TRIAL

#### 7.2.2.46. Function: id

Returns the id of a relationship.

TRIAL

#### 7.2.2.47. Function: isEmpty

Checks whether a list is empty.

## Input/Output

#### 7.2.2.48. Function: isEmpty

Checks whether a map is empty.

TRIAL

#### 7.2.2.49. Function: isEmpty

Checks whether a string is empty.

TRIAL

### 7.2.2.50. Function: keys

Returns a list containing the string representations for all the property names of a node.

#### Input/Output

	Name	Data type	Description
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	MAP?	
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	MAP?	
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	MAP?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	
*@	Returns	LIST? OF STRING?	

### 7.2.2.51. Function: keys

Returns a list containing the string representations for all the property names of a relationship

TRIAL

### 7.2.2.52. Function: keys

Returns a list containing the string representations for all the property names of a map.

TRIAL

### 7.2.2.53. Function: labels

Returns a list containing the string representations for all the labels of a node.

Input/Output

	Name	Data type	Description
→@	input	NODE?	
*@	Returns	LIST? OF STRING?	

TRIAL

#### 7.2.2.54. Function: last

Returns the last element in a list.

##### Input/Output

	Name	Data type	Description
@	list	LIST? OF ANY?	
*@	Returns	ANY?	

TRIAL

### 7.2.2.55. Function: left

Returns a string containing the specified number of leftmost characters of the original string.

#### Input/Output

	Name	Data type	Description
@	original	STRING?	
@	length	INTEGER?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.56. Function: length

Returns the length of a path.

Input/Output

	Name	Data type	Description
@	input	PATH?	
*@	Returns	INTEGER?	

TRIAL

### 7.2.2.57. Function: linenumber

Returns the line number that LOAD CSV is currently using.

Input/Output

	Name	Data type	Description
*@	Returns	INTEGER?	

TRIAL

### 7.2.2.58. Function: localdatetime

Create a LocalDateTime instant.

Input/Output

	Name	Data type	Description
→@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALDATETIME?	

TRIAL

### 7.2.2.59. Function: localdatetime.realtime

Get the current LocalDateTime instant using the realtime clock.

#### Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALDATETIME?	

TRIAL

### 7.2.2.60. Function: localdatetime.statement

Get the current LocalDateTime instant using the statement clock.

#### Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALDATETIME?	

TRIAL

### 7.2.2.61. Function: localdatetime.transaction

Get the current LocalDateTime instant using the transaction clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALDATETIME?	

TRIAL

### 7.2.2.62. Function: localdatetime.truncate

Truncate the input temporal value to a LocalDateTime instant using the specified unit.

#### Input/Output

	Name	Data type	Description
@	unit	STRING?	
@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
@	fields = null	MAP?	
*	Returns	LOCALDATETIME?	

TRIAL

### 7.2.2.63. Function: localtime

Create a LocalTime instant.

Input/Output

	Name	Data type	Description
→@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALTIME?	

TRIAL

#### 7.2.2.64. Function: localtime.realtime

Get the current LocalTime instant using the realtime clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALTIME?	

TRIAL

### 7.2.2.65. Function: localtime.statement

Get the current LocalTime instant using the statement clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALTIME?	

TRIAL

### 7.2.2.66. Function: localtime.transaction

Get the current LocalTime instant using the transaction clock.

#### Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	LOCALTIME?	

TRIAL

### 7.2.2.67. Function: localtime.truncate

Truncate the input temporal value to a LocalTime instant using the specified unit.

#### Input/Output

	Name	Data type	Description
@	unit	STRING?	
@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
@	fields = null	MAP?	
*@	Returns	LOCALTIME?	

TRIAL

### 7.2.2.68. Function: log

Returns the natural logarithm of a number.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.69. Function: log10

Returns the common logarithm (base 10) of a number.

Input/Output

	Name	Data type	Description
→@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.70. Function: ltrim

Returns the original string with leading whitespace removed.

#### Input/Output

	Name	Data type	Description
@	input	STRING?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.71. Function: max

Returns the maximum value in a set of values.

#### Input/Output

	Name	Data type	Description
@	input	ANY?	
*@	Returns	ANY?	

TRIAL

### 7.2.2.72. Function: min

Returns the minimum value in a set of values.

#### Input/Output

	Name	Data type	Description
>@	input	ANY?	
*@	Returns	ANY?	

TRIAL

### 7.2.2.73. Function: nodes

Returns a list containing all the nodes in a path.

Input/Output

	Name	Data type	Description
@	input	PATH?	
*@	Returns	LIST? OF NODE?	

TRIAL

#### 7.2.2.74. Function: none

Returns true if the predicate holds for no element in the given list.

Input/Output

	Name	Data type	Description
*@	Returns	BOOLEAN?	

TRIAL

### 7.2.2.75. Function: percentileCont

Returns the percentile of a value over a group using linear interpolation.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
@	percentile	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.76. Function: percentileDisc

Returns the nearest integer value to the given percentile over a group using a rounding method.

#### Input/Output

	Name	Data type	Description
@	input	INTEGER?	
@	input	FLOAT?	
@	input	INTEGER?	
@	input	FLOAT?	
@	percentile	FLOAT?	
@	Returns	INTEGER?	
@	Returns	FLOAT?	
@	Returns	INTEGER?	
@	Returns	FLOAT?	

#### 7.2.2.77. Function: percentileDisc

Returns the nearest floating point value to the given percentile over a group using a rounding method.

TRIAL

### 7.2.2.78. Function: pi

Returns the mathematical constant pi.

Input/Output

	Name	Data type	Description
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.79. Function: point

Returns a 2D or 3D point object, given two or respectively three coordinate values in the Cartesian coordinate system or WGS 84 geographic coordinate system.

#### Input/Output

	Name	Data type	Description
→@	input	MAP?	
*@	Returns	POINT?	

TRIAL

### 7.2.2.80. Function: properties

Returns a map containing all the properties of a node.

#### Input/Output

	Name	Data type	Description
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	MAP?	
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	MAP?	
@@	input	NODE?	
@@	input	RELATIONSHIP?	
@@	input	MAP?	
*@	Returns	MAP?	

#### 7.2.2.81. Function: properties

Returns a map containing all the properties of a relationship.

TRIAL

#### 7.2.2.82. Function: properties

Returns a map containing all the properties of a map.

TRIAL

### 7.2.2.83. Function: radians

Converts degrees to radians.

Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

#### 7.2.2.84. Function: rand

Returns a random floating point number in the range from 0 (inclusive) to 1 (exclusive); i.e. [0,1).

##### Input/Output

Name	Data type	Description
*@ Returns	FLOAT?	

TRIAL

### 7.2.2.85. Function: randomUUID

Generates a random UUID.

Input/Output

	Name	Data type	Description
*@	Returns	STRING?	

TRIAL

### 7.2.2.86. Function: range

Returns a list comprising all integer values within a specified range.

#### Input/Output

	Name	Data type	Description
@	start	INTEGER?	
@	end	INTEGER?	
*	Returns	LIST? OF INTEGER?	
*	step	INTEGER?	
*	Returns	LIST? OF INTEGER?	
*	step	INTEGER?	
*	Returns	LIST? OF INTEGER?	
*	Returns	LIST? OF INTEGER?	

### 7.2.2.87. Function: range

Returns a list comprising all integer values within a specified range created with step length.

TRIAL

### 7.2.2.88. Function: reduce

Runs an expression against individual elements of a list, storing the result of the expression in an accumulator.

#### Input/Output

Name	Data type	Description
*@ Returns	ANY?	

TRIAL

### 7.2.2.89. Function: relationships

Returns a list containing all the relationships in a path.

Input/Output

	Name	Data type	Description
@	input	PATH?	
*@	Returns	LIST? OF RELATIONSHIP?	

TRIAL

### 7.2.2.90. Function: replace

Returns a string in which all occurrences of a specified search string in the original string have been replaced by another (specified) replace string.

#### Input/Output

	Name	Data type	Description
@	original	STRING?	
@	search	STRING?	
@	replace	STRING?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.91. Function: reverse

Returns a string in which the order of all characters in the original string have been reversed.

#### Input/Output

	Name	Data type	Description
>@	input	STRING?	
>@	input	LIST? OF ANY?	
>@	input	STRING?	
>@	input	LIST? OF ANY?	
*@	Returns	STRING?	
*@	Returns	LIST? OF ANY?	
*@	Returns	STRING?	
*@	Returns	LIST? OF ANY?	

TRIAL

### 7.2.2.92. Function: reverse

Returns a list in which the order of all elements in the original list have been reversed.

TRIAL

### 7.2.2.93. Function: right

Returns a string containing the specified number of rightmost characters of the original string.

#### Input/Output

	Name	Data type	Description
@	original	STRING?	
@	length	INTEGER?	
*@	Returns	STRING?	



TRIAL

### 7.2.2.94. Function: round

Returns the value of a number rounded to the nearest integer.

#### Input/Output

	Name	Data type	Description
>@	input	FLOAT?	
>@	value	FLOAT?	
@@	value	FLOAT?	
@@	input	FLOAT?	
@@	value	FLOAT?	
@@	value	FLOAT?	
@@	input	FLOAT?	
@@	value	FLOAT?	
@@	value	FLOAT?	
*@	Returns	FLOAT?	
*@	precision	NUMBER?	
*@	precision	NUMBER?	
*@	Returns	FLOAT?	
*@	precision	NUMBER?	
*@	precision	NUMBER?	
*@	Returns	FLOAT?	
*@	precision	NUMBER?	
*@	precision	NUMBER?	
*@	Returns	FLOAT?	
*@	mode	STRING?	
*@	Returns	FLOAT?	
*@	mode	STRING?	
*@	Returns	FLOAT?	
*@	mode	STRING?	
*@	Returns	FLOAT?	
*@	Returns	FLOAT?	
*@	Returns	FLOAT?	

### 7.2.2.95. Function: round

Returns the value of a number rounded to the specified precision using rounding mode HALF\_UP.

TRIAL

### 7.2.2.96. Function: round

Returns the value of a number rounded to the specified precision with the specified rounding mode.

TRIAL

### 7.2.2.97. Function: rtrim

Returns the original string with trailing whitespace removed.

#### Input/Output

	Name	Data type	Description
@	input	STRING?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.98. Function: sign

Returns the signum of an integer number: 0 if the number is 0, -1 for any negative number, and 1 for any positive number.

#### Input/Output

	Name	Data type	Description
@	input	INTEGER?	
@	input	FLOAT?	
@	input	INTEGER?	
@	input	FLOAT?	
*	Returns	INTEGER?	

TRIAL

### 7.2.2.99. Function: sign

Returns the signum of a floating point number: 0 if the number is 0, -1 for any negative number, and 1 for any positive number.

TRIAL

### 7.2.2.100. Function: sin

Returns the sine of a number.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.101. Function: single

Returns true if the predicate holds for exactly one of the elements in the given list.

#### Input/Output

	Name	Data type	Description
*@	Returns	BOOLEAN?	

TRIAL

### 7.2.2.102. Function: size

Returns the number of items in a list.

#### Input/Output

	Name	Data type	Description
>@	input	LIST? OF ANY?	
>@	input	STRING?	
>@	input	LIST? OF ANY?	
>@	input	STRING?	
*@	Returns	INTEGER?	

TRIAL

### 7.2.2.103. Function: size

Returns the number of Unicode characters in a string.

TRIAL

#### 7.2.2.104. Function: split

Returns a list of strings resulting from the splitting of the original string around matches of the given delimiter.

##### Input/Output

	Name	Data type	Description
@	original	STRING?	
@	splitDelimiter	STRING?	
@	splitDelimiters	LIST? OF STRING?	
@	splitDelimiter	STRING?	
@	splitDelimiters	LIST? OF STRING?	
@	Returns	LIST? OF STRING?	
@	Returns	LIST? OF STRING?	
@	Returns	LIST? OF STRING?	
@	Returns	LIST? OF STRING?	

### 7.2.2.105. Function: split

Returns a list of strings resulting from the splitting of the original string around matches of any of the given delimiters.

TRIAL

### 7.2.2.106. Function: sqrt

Returns the square root of a number.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.107. Function: startNode

Returns the start node of a relationship.

#### Input/Output

	Name	Data type	Description
>@	input	RELATIONSHIP?	
*@	Returns	NODE?	

TRIAL

### 7.2.2.108. Function: stdev

Returns the standard deviation for the given value over a group for a sample of a population.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.109. Function: stdevp

Returns the standard deviation for the given value over a group for an entire population.

#### Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.110. Function: substring

Returns a substring of the original string, beginning with a 0-based index start.

#### Input/Output

	Name	Data type	Description
@	original	STRING?	
@	start	INTEGER?	
*	Returns	STRING?	
@	length	INTEGER?	
*	Returns	STRING?	
@	length	INTEGER?	
*	Returns	STRING?	
*	Returns	STRING?	

### 7.2.2.111. Function: substring

Returns a substring of length 'length' of the original string, beginning with a 0-based index start.

TRIAL

### 7.2.2.112. Function: sum

Returns the sum of a set of integers

Input/Output

	Name	Data type	Description
@@	input	INTEGER?	
@@	input	FLOAT?	
@@	input	DURATION?	
@@	input	INTEGER?	
@@	input	FLOAT?	
@@	input	DURATION?	
@@	input	INTEGER?	
@@	input	FLOAT?	
@@	input	DURATION?	
*@	Returns	INTEGER?	
*@	Returns	FLOAT?	
*@	Returns	DURATION?	
*@	Returns	INTEGER?	
*@	Returns	FLOAT?	
*@	Returns	DURATION?	
*@	Returns	INTEGER?	
*@	Returns	FLOAT?	
*@	Returns	DURATION?	

### 7.2.2.113. Function: sum

Returns the sum of a set of floats

TRIAL

#### 7.2.2.114. Function: sum

Returns the sum of a set of durations

TRIAL

### 7.2.2.115. Function: tail

Returns all but the first element in a list.

#### Input/Output

	Name	Data type	Description
@	input	LIST? OF ANY?	
*@	Returns	LIST? OF ANY?	

TRIAL

### 7.2.2.116. Function: tan

Returns the tangent of a number.

Input/Output

	Name	Data type	Description
@	input	FLOAT?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.117. Function: time

Create a Time instant.

Input/Output

	Name	Data type	Description
→@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	TIME?	

TRIAL

### 7.2.2.118. Function: time.realtime

Get the current Time instant using the realtime clock.

#### Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	TIME?	

TRIAL

### 7.2.2.119. Function: time.statement

Get the current Time instant using the statement clock.

#### Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	TIME?	

TRIAL

### 7.2.2.120. Function: time.transaction

Get the current Time instant using the transaction clock.

Input/Output

	Name	Data type	Description
→@	timezone = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
*@	Returns	TIME?	

TRIAL

### 7.2.2.121. Function: time.truncate

Truncate the input temporal value to a Time instant using the specified unit.

#### Input/Output

	Name	Data type	Description
@	unit	STRING?	
@	input = DEFAULT_TEMPORAL_ARGUMENT	ANY?	
@	fields = null	MAP?	
*	Returns	TIME?	

TRIAL

#### 7.2.2.122. Function: toBoolean

Converts a string value to a boolean value.

## Input/Output

### 7.2.2.123. Function: toBoolean

Converts a boolean value to a boolean value.

TRIAL

#### 7.2.2.124. Function: toBoolean

Converts a integer value to a boolean value. 0 is defined to be FALSE and any other integer is defined to be TRUE.

TRIAL

### 7.2.2.125. Function: toBooleanList

Converts a list of values to a list of boolean values. If any values are not convertible to boolean they will be null in the list returned.

#### Input/Output

	Name	Data type	Description
→@	input	LIST? OF ANY?	
*@	Returns	LIST? OF BOOLEAN?	

TRIAL

### 7.2.2.126. Function: toBooleanOrNull

Converts a value to a boolean value, or null if the value cannot be converted.

#### Input/Output

	Name	Data type	Description
@	input	ANY?	
*@	Returns	BOOLEAN?	

TRIAL

### 7.2.2.127. Function: toFloat

Converts a string value to a floating point value.

#### Input/Output

	Name	Data type	Description
@	input	STRING?	
@	input	NUMBER?	
@	input	STRING?	
@	input	NUMBER?	
*	Returns	FLOAT?	

TRIAL

### 7.2.2.128. Function: toFloat

Converts an integer value to a floating point value.

TRIAL

### 7.2.2.129. Function: toFloatList

Converts a list of values to a list of float values. If any values are not convertible to float they will be null in the list returned.

#### Input/Output

	Name	Data type	Description
→@	input	LIST? OF ANY?	
*@	Returns	LIST? OF FLOAT?	

TRIAL

### 7.2.2.130. Function: toFloatOrNull

Converts a value to a floating point value, or null if the value cannot be converted.

#### Input/Output

	Name	Data type	Description
@	input	ANY?	
*@	Returns	FLOAT?	

TRIAL

### 7.2.2.131. Function: tolnteger

Converts a string value to an integer value.

#### Input/Output

	Name	Data type	Description
@@	input	STRING?	
@@	input	NUMBER?	
@@	input	BOOLEAN?	
@@	input	STRING?	
@@	input	NUMBER?	
@@	input	BOOLEAN?	
@@	input	STRING?	
@@	input	NUMBER?	
@@	input	BOOLEAN?	
*@	Returns	INTEGER?	

### 7.2.2.132. Function: tolnteger

Converts a floating point value to an integer value.

TRIAL

### 7.2.2.133. Function: tolnteger

Converts a boolean to an integer value. TRUE is defined to be 1 and FALSE is defined to be 0.

TRIAL

#### 7.2.2.134. Function: tolntegerList

Converts a list of values to a list of integer values. If any values are not convertible to integer they will be null in the list returned.

##### Input/Output

	Name	Data type	Description
→@	input	LIST? OF ANY?	
*@	Returns	LIST? OF INTEGER?	

TRIAL

### 7.2.2.135. Function: tolntegerOrNull

Converts a value to an integer value, or null if the value cannot be converted.

#### Input/Output

	Name	Data type	Description
@	input	ANY?	
*@	Returns	INTEGER?	

TRIAL

### 7.2.2.136. Function: toLower

Returns the original string in lowercase.

#### Input/Output

	Name	Data type	Description
@	input	STRING?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.137. Function: `toString`

Converts an integer, float, boolean, point or temporal type (i.e. Date, Time, LocalTime, DateTime, LocalDateTime or Duration) value to a string.

#### Input/Output

	Name	Data type	Description
→@	input	ANY?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.138. Function: `toStringList`

Converts a list of values to a list of string values. If any values are not convertible to string they will be null in the list returned.

#### Input/Output

	Name	Data type	Description
→@	input	LIST? OF ANY?	
*@	Returns	LIST? OF STRING?	

TRIAL

### 7.2.2.139. Function: `toStringOrNull`

Converts an integer, float, boolean, point or temporal type (i.e. Date, Time, LocalTime, DateTime, LocalDateTime or Duration) value to a string, or null if the value cannot be converted.

#### Input/Output

	Name	Data type	Description
→@	input	ANY?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.140. Function: toUpper

Returns the original string in uppercase.

#### Input/Output

	Name	Data type	Description
@	input	STRING?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.141. Function: trim

Returns the original string with leading and trailing whitespace removed.

#### Input/Output

	Name	Data type	Description
@	input	STRING?	
*@	Returns	STRING?	

TRIAL

### 7.2.2.142. Function: type

Returns the string representation of the relationship type.

Input/Output

	Name	Data type	Description
>@	input	RELATIONSHIP?	
*@	Returns	STRING?	

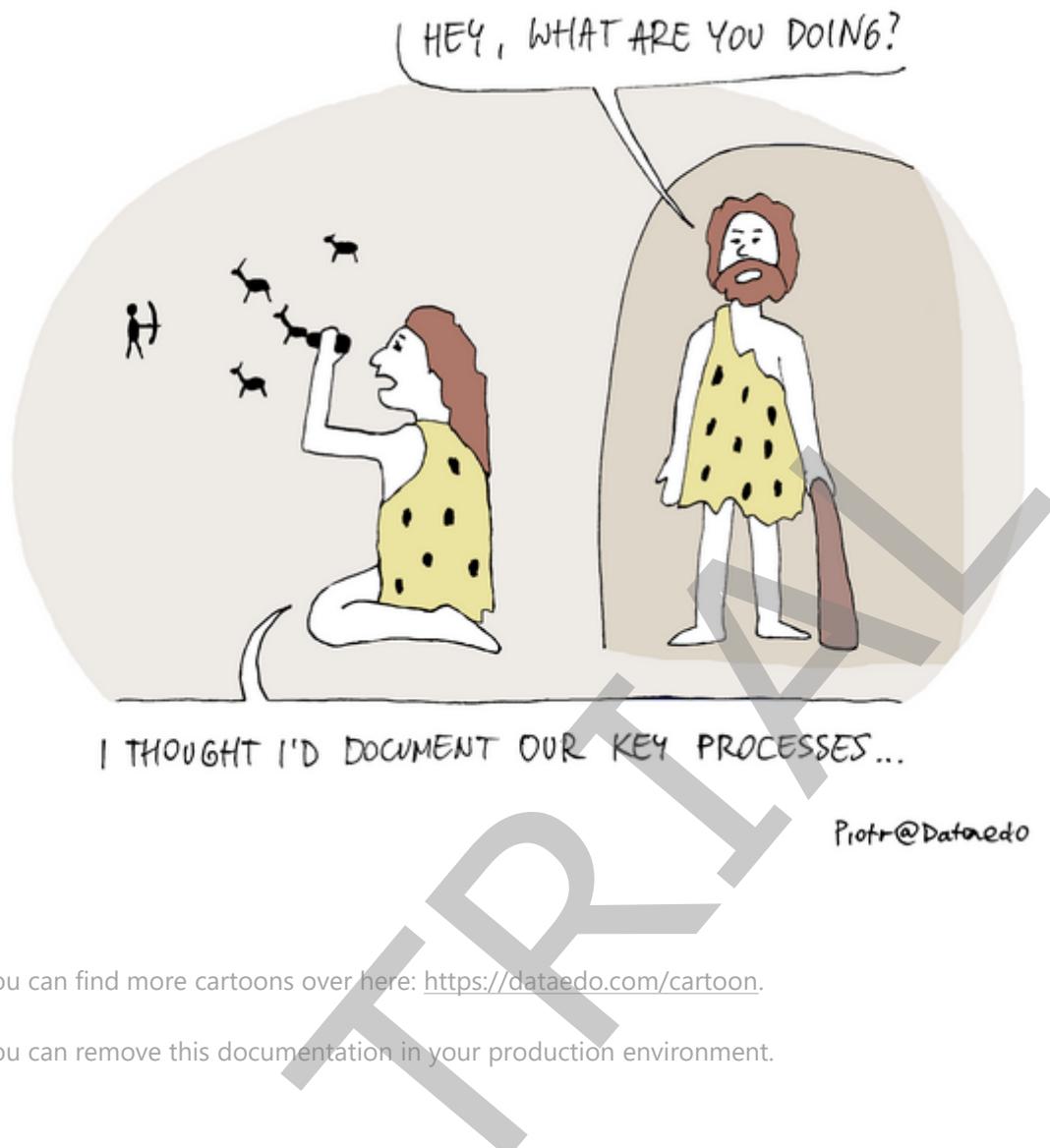
TRIAL

TRIAL

## 8. Sample SQL Database

This is a documentation of a sample **SQL** database.

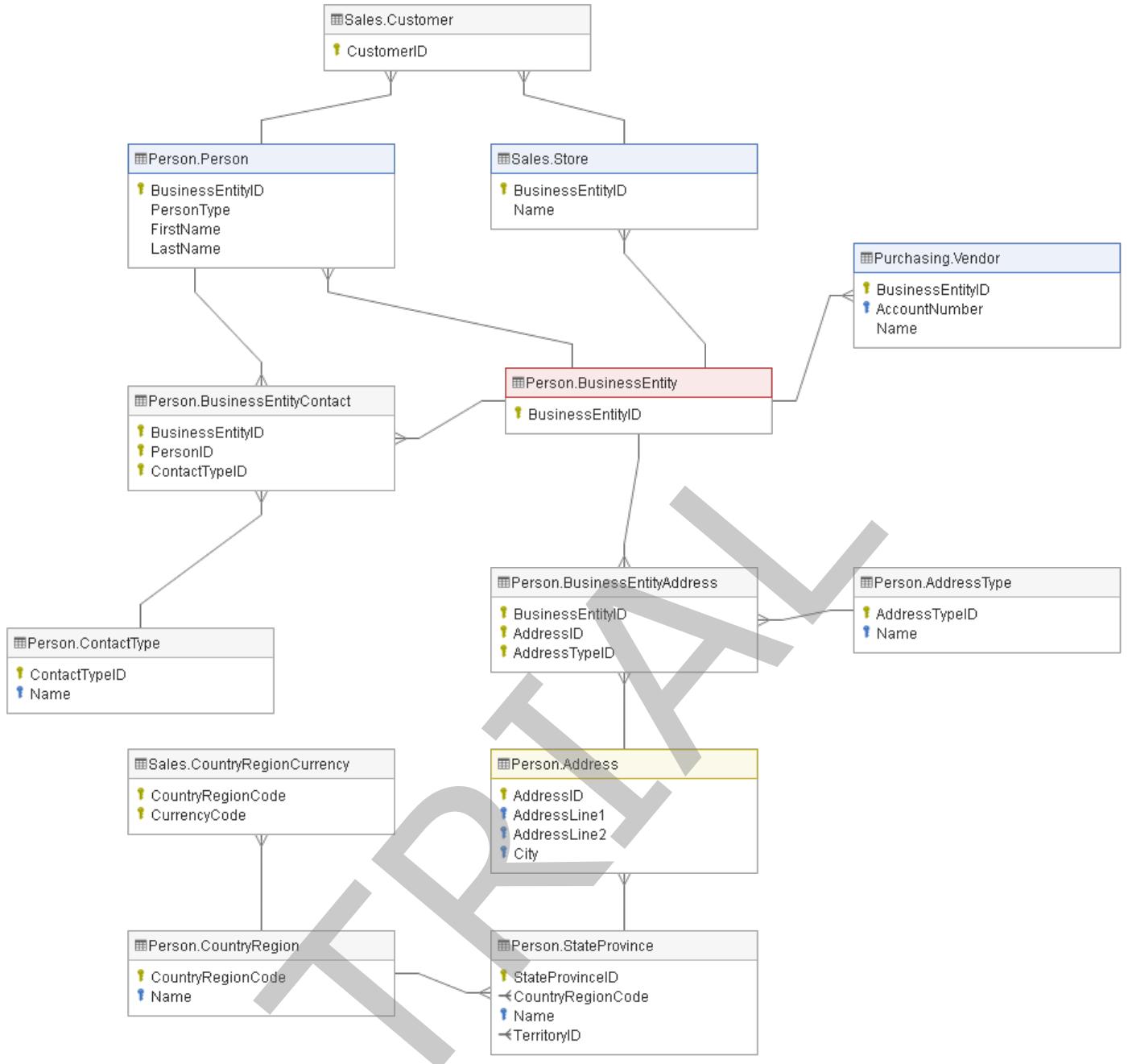
And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

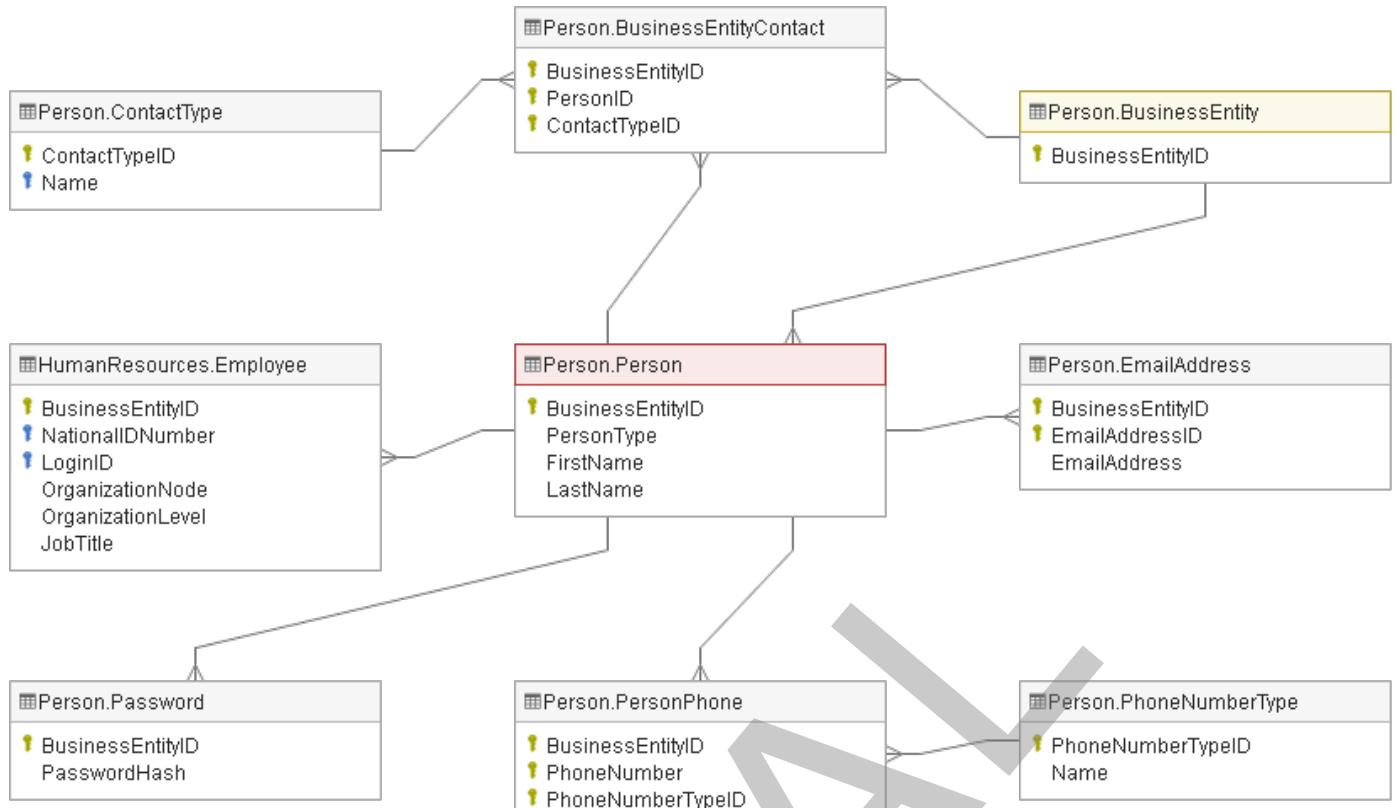
## 8.1. Business Entities



Vendors, customers, and employees have common tables for addresses and contacts. Those tables are linked to one table - **BusinessEntity** that holds ID for all vendors, customers, and employees tables.

TRIAL

## 8.2. People



Names and addresses of individual customers, vendors, and employees.

## 8.2.1. Tables

### 8.2.1.1. Table: Person.Address

**Sample field: Status:** Active

Street address information for customers, employees, and vendors.

#### Columns

		Name	Data type	Description / Attributes
AddressID	Primary key for Address records. <b>Identity / Auto increment</b>	AddressID	int	Primary key for Address records. <b>Identity / Auto increment</b>
AddressLine1	First street address line.	AddressLine1	nvarchar(60)	First street address line.
AddressLine2	Second street address line. <b>Nullable</b>	AddressLine2	nvarchar(60)	Second street address line. <b>Nullable</b>
City	Name of the city.	City	nvarchar(30)	Name of the city.
StateProvinceID	Unique identification number for the state or province. Foreign key to StateProvince table. <b>References:</b> Person.StateProvince	StateProvinceID	int	Unique identification number for the state or province. Foreign key to StateProvince table. <b>References:</b> Person.StateProvince
PostalCode	Postal code for the street address.	PostalCode	nvarchar(15)	Postal code for the street address.
SpatialLocation	Latitude and longitude of this address. <b>Nullable</b>	SpatialLocation	geography	Latitude and longitude of this address. <b>Nullable</b>
rowguid	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
ModifiedDate	Date and time the record was last updated. <b>Default:</b> getdate()	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
Person.StateProvince		<b>Person.Address</b> StateProvinceID = Person.StateProvinceStateProvinceID	FK_Address_StateProvince_StateProvinceID Foreign key constraint referencing StateProvince.StateProvinceID.

#### Linked from

Table		Join	Title / Name / Description
Person.BusinessEntityAddress		<b>Person.Address</b> AddressID = Person.BusinessEntityAddressAddressID	FK_BusinessEntityAddress_Address_AddressID Foreign key constraint referencing Address.AddressID.
Sales.SalesOrderHeader		<b>Person.Address</b> AddressID = Sales.SalesOrderHeaderBillToAddressID	FK_SalesOrderHeader_Address_BillToAddressID Foreign key constraint referencing Address.AddressID.
Sales.SalesOrderHeader		<b>Person.Address</b> AddressID = Sales.SalesOrderHeaderShipToAddressID	FK_SalesOrderHeader_Address_ShipToAddressID Foreign key constraint referencing Address.AddressID.

#### Unique keys

Columns		Name / Description
AddressID	PK_Address_AddressID Primary key (clustered) constraint	PK_Address_AddressID Primary key (clustered) constraint
rowguid	AK_Address_rowguid Unique nonclustered index. Used to support replication samples.	AK_Address_rowguid Unique nonclustered index. Used to support replication samples.

Columns	Name / Description
 AddressLine1, AddressLine2, City, StateProvinceID, PostalCode	IX_Address_AddressLine1_AddressLine2_City_StateProvinceID_PostalCode Nonclustered index.

## Uses

Name
 Person.Address
→ Person.StateProvince

## Used By

Name
 Person.Address
↳ HumanResources.vEmployee
↳ Purchasing.vVendorWithAddresses
↳ Sales.vIndividualCustomer
↳ Sales.vSalesPerson
↳ Sales.vStoreWithAddresses
← Person.BusinessEntityAddress
← Sales.SalesOrderHeader
← Sales.SalesOrderHeader

## 8.2.1.2. Table: Person.AddressType

**Sample field: Status:** Active

Types of addresses stored in the Address table.

### Columns

		Name	Data type	Description / Attributes
■	🔑	AddressTypeID	int	Primary key for AddressType records. <b>Identity / Auto increment</b>
■	🔑	Name	Name: nvarchar(50)	Address type description. For example, Billing, Home, or Shipping.
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
→	Person.BusinessEntityAddress	Person.AddressTypeAddressTypeID = Person.BusinessEntityAddressAddressTypeID	FK_BusinessEntityAddress_AddressType_AddressTypeID Foreign key constraint referencing AddressType.AddressTypeID.

### Unique keys

Columns		Name / Description
🔑	AddressTypeID	PK_AddressType_AddressTypeID Primary key (clustered) constraint
🔑	Name	AK_AddressType_Name Unique nonclustered index.
🔑	rowguid	AK_AddressType_rowguid Unique nonclustered index. Used to support replication samples.

### Used By

		Name
■	Person.AddressType	
↳	Purchasing.vVendorWithAddresses	
↳	Sales.vIndividualCustomer	
↳	Sales.vStoreWithAddresses	
→	Person.BusinessEntityAddress	

### 8.2.1.3. Table: Person.BusinessEntity

**Sample field: Status:** Active

Source of the ID that connects vendors, customers, and employees with address and contact information.

#### Columns

		Name	Data type	Description / Attributes
		BusinessEntityID	int	Primary key for all customers, vendors, and employees. <b>Identity / Auto increment</b>
		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

	Table	Join	Title / Name / Description
→	Person.BusinessEntityAddress	<b>Person.BusinessEntity</b> BusinessEntityID = Person.BusinessEntityAddressBusinessEntityID	FK_BusinessEntityAddress_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID.
→	Person.BusinessEntityContact	<b>Person.BusinessEntity</b> BusinessEntityID = Person.BusinessEntityContactBusinessEntityID	FK_BusinessEntityContact_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID.
→	Person.Person	<b>Person.BusinessEntity</b> BusinessEntityID = Person.PersonBusinessEntityID	FK_Person_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID.
→	Sales.Store	<b>Person.BusinessEntity</b> BusinessEntityID = Sales.StoreBusinessEntityID	FK_Store_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID
→	Purchasing.Vendor	<b>Person.BusinessEntity</b> BusinessEntityID = Purchasing.VendorBusinessEntityID	FK_Vendor_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID

#### Unique keys

	Columns	Name / Description
	BusinessEntityID	PK_BusinessEntity_BusinessEntityID Primary key (clustered) constraint
	rowguid	AK_BusinessEntity_rowguid Unique nonclustered index. Used to support replication samples.

#### Used By

	Name
	<b>Person.BusinessEntity</b>
→	Person.BusinessEntityAddress
→	Person.BusinessEntityContact
→	Person.Person
→	Purchasing.Vendor
→	Sales.Store

## 8.2.1.4. Table: Person.BusinessEntityAddress

**Sample field: Status:** Active

Cross-reference table mapping customers, vendors, and employees to their addresses.

### Columns

		Name	Data type	Description / Attributes
■	🔑	BusinessEntityID	int	Primary key. Foreign key to BusinessEntity.BusinessEntityID. <b>References:</b> Person.BusinessEntity
■	🔑	AddressID	int	Primary key. Foreign key to Address.AddressID. <b>References:</b> Person.Address
■	🔑	AddressTypeID	int	Primary key. Foreign key to AddressType.AddressTypeID. <b>References:</b> Person.AddressType
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.Address	<b>Person.BusinessEntityAddress</b> AddressID = Person.AddressAddressID	FK_BusinessEntityAddress_Address_AddressID Foreign key constraint referencing Address.AddressID.
→	Person.AddressType	<b>Person.BusinessEntityAddress</b> AddressTypeID = Person.AddressTypeAddressTypeID	FK_BusinessEntityAddress_AddressType_AddressTypeID Foreign key constraint referencing AddressType.AddressTypeID.
→	Person.BusinessEntity	<b>Person.BusinessEntityAddress</b> BusinessEntityID = Person.BusinessEntityBusinessEntityID	FK_BusinessEntityAddress_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID, AddressID, AddressTypeID	PK_BusinessEntityAddress_BusinessEntityID_AddressID_AddressTypeID Primary key (clustered) constraint
🔑	rowguid	AK_BusinessEntityAddress_rowguid Unique nonclustered index. Used to support replication samples.

### Uses

Name	
■	<b>Person.BusinessEntityAddress</b>
→	Person.Address
→	Person.AddressType
→	Person.BusinessEntity

### Used By

Name	
■	<b>Person.BusinessEntityAddress</b>
↳	HumanResources.vEmployee

Name

Purchasing.vVendorWithAddresses

Sales.vIndividualCustomer

Sales.vSalesPerson

Sales.vStoreWithAddresses

TRIAL

## 8.2.1.5. Table: Person.BusinessEntityContact

**Sample field: Status:** Active

Cross-reference table mapping stores, vendors, and employees to people

### Columns

Name			Data type	Description / Attributes
BusinessEntityID	key	BusinessEntityID	int	Primary key. Foreign key to BusinessEntity.BusinessEntityID. <b>References:</b> Person.BusinessEntity
PersonID	key	PersonID	int	Primary key. Foreign key to Person.BusinessEntityID. <b>References:</b> Person.Person
ContactTypeID	key	ContactTypeID	int	Primary key. Foreign key to ContactType.ContactTypeID. <b>References:</b> Person.ContactType
rowguid	key	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
ModifiedDate		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

	Table	Join	Title / Name / Description
→	Person.BusinessEntity	<b>Person.BusinessEntityContact</b> BusinessEntityID = Person.BusinessEntityBusinessEntityID	FK_BusinessEntityContact_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID.
→	Person.ContactType	<b>Person.BusinessEntityContact</b> ContactTypeID = Person.ContactTypeContactTypeID	FK_BusinessEntityContact_ContactType_ContactTypeID Foreign key constraint referencing ContactType.ContactTypeID.
→	Person.Person	<b>Person.BusinessEntityContact</b> PersonID = Person.PersonBusinessEntityID	FK_BusinessEntityContact_Person_PersonID Foreign key constraint referencing Person.BusinessEntityID.

### Unique keys

	Columns	Name / Description
key	BusinessEntityID, PersonID, ContactTypeID	PK_BusinessEntityContact_BusinessEntityID_PersonID_ContactTypeID Primary key (clustered) constraint
key	rowguid	AK_BusinessEntityContact_rowguid Unique nonclustered index. Used to support replication samples.

### Uses

	Name
Person.BusinessEntityContact	
→ Person.BusinessEntity	
→ Person.ContactType	
→ Person.Person	

### Used By

	Name
Person.BusinessEntityContact	
↳ Purchasing.vVendorWithContacts	

Name

 Sales.vStoreWithContacts

 dbo.ufnGetContactInformation

TRIAL

## 8.2.1.6. Table: Person.ContactType

**Sample field: Status:** Active

Lookup table containing the types of business entity contacts.

### Columns

		Name	Data type	Description / Attributes
		ContactTypeID	int	Primary key for ContactType records. <b>Identity / Auto increment</b>
		Name	Name: nvarchar(50)	Contact type description.
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
	Person.BusinessEntityContact	<b>Person.ContactType</b> ContactTypeID = Person.BusinessEntityContactContactTypeID	FK_BusinessEntityContact_ContactType_ContactTypeID Foreign key constraint referencing ContactType.ContactTypeID.

### Unique keys

Columns		Name / Description
	ContactTypeID	PK_ContactType_ContactTypeID Primary key (clustered) constraint
	Name	AK_ContactType_Name Unique nonclustered index.

### Used By

		Name
	<b>Person.ContactType</b>	
	Purchasing.vVendorWithContacts	
	Sales.vStoreWithContacts	
	dbo.ufnGetContactInformation	
	Person.BusinessEntityContact	

## 8.2.1.7. Table: Person.CountryRegion

**Sample field: Status:** Active

Lookup table containing the ISO standard codes for countries and regions.

### Columns

Name		Data type	Description / Attributes
CountryRegionCode		nvarchar(3)	ISO standard code for countries and regions.
Name		Name: nvarchar(50)	Country or region name.
ModifiedDate		datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

	Table	Join	Title / Name / Description
→	Sales.CountryRegionCurrency	<b>Person.CountryRegion</b> CountryRegion Code = Sales.CountryRegionCurrencyCountryRegionCode	FK_CountryRegionCurrency_CountryRegion_CountryRegionCode Foreign key constraint referencing CountryRegion.CountryRegionCode.
→	Sales.SalesTerritory	<b>Person.CountryRegion</b> CountryRegion Code = Sales.SalesTerritoryCountryRegionCode	FK_SalesTerritory_CountryRegion_CountryRegionCode Foreign key constraint referencing CountryRegion.CountryRegionCode.
→	Person.StateProvince	<b>Person.CountryRegion</b> CountryRegion Code = Person.StateProvinceCountryRegionCode	FK_StateProvince_CountryRegion_CountryRegionCode Foreign key constraint referencing CountryRegion.CountryRegionCode.

### Unique keys

	Columns	Name / Description
	CountryRegionCode	PK_CountryRegion_CountryRegionCode Primary key (clustered) constraint
	Name	AK_CountryRegion_Name Unique nonclustered index.

### Used By

	Name
	<b>Person.CountryRegion</b>
	HumanResources.vEmployee
	Person.vStateProvinceCountryRegion
	Purchasing.vVendorWithAddresses
	Sales.vIndividualCustomer
	Sales.vSalesPerson
	Sales.vStoreWithAddresses
→	Person.StateProvince
→	Sales.CountryRegionCurrency
→	Sales.SalesTerritory

## 8.2.1.8. Table: Person.EmailAddress

**Sample field: Status:** Active

Where to send a person email.

### Columns

		Name	Data type	Description / Attributes
☰	🔑	BusinessEntityID	int	Primary key. Person associated with this email address. Foreign key to Person.BusinessEntityID <b>References:</b> Person.Person
☰	🔑	EmailAddressID	int	Primary key. ID of this email address. <b>Identity / Auto increment</b>
☰		EmailAddress	nvarchar(50)	E-mail address for the person. <b>Nullable</b>
☰		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
☰		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.Person	Person.EmailAddressBusinessEntityID = Person.PersonBusinessEntityID	FK_EmailAddress_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID, EmailAddressID	PK_EmailAddress_BusinessEntityID_EmailAddressID Primary key (clustered) constraint

### Uses

Name	
☰	Person.EmailAddress
→	Person.Person

### Used By

Name	
☰	Person.EmailAddress
↳	HumanResources.vEmployee
↳	Purchasing.vVendorWithContacts
↳	Sales.vIndividualCustomer
↳	Sales.vSalesPerson
↳	Sales.vStoreWithContacts

## 8.2.1.9. Table: Person.Password

**Sample field: Status:** Active

One way hashed authentication information

### Columns

		Name	Data type	Description / Attributes
☰	🔑	BusinessEntityID	int	Person identification number. Foreign key to Person.BusinessEntityID. <b>References:</b> Person.Person
☰		PasswordHash	varchar(128)	Password for the e-mail account.
☰		PasswordSalt	varchar(10)	Random value concatenated with the password string before the password is hashed.
☰		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
☰		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.Person	Person.PasswordBusinessEntityID = Person.PersonBusinessEntityID	FK_Password_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID	PK_Password_BusinessEntityID Primary key (clustered) constraint

### Uses

Name	
☰	Person.Password
→	Person.Person

## 8.2.1.10. Table: Person.Person

**Sample field: Status:** Active

Human beings involved with AdventureWorks: employees, customer contacts, and vendor contacts.

### Columns

	Name	Data type	Description / Attributes
PK	BusinessEntityID	int	Primary key for Person records. <b>References:</b> Person.BusinessEntity
	PersonType	nchar(2)	Primary type of person: SC = Store Contact, IN = Individual (retail) customer, SP = Sales person, EM = Employee (non-sales), VC = Vendor contact, GC = General contact
	NameStyle	NameStyle: bit	0 = The data in FirstName and LastName are stored in western style (first name, last name) order. 1 = Eastern style (last name, first name) order. <b>Default:</b> 0
	Title	nvarchar(8)	A courtesy title. For example, Mr. or Ms. <b>Nullable</b>
	FirstName	Name: nvarchar(50)	First name of the person.
	MiddleName	Name: nvarchar(50)	Middle name or middle initial of the person. <b>Nullable</b>
	LastName	Name: nvarchar(50)	Last name of the person.
	Suffix	nvarchar(10)	Surname suffix. For example, Sr. or Jr. <b>Nullable</b>
	EmailPromotion	int	0 = Contact does not wish to receive e-mail promotions, 1 = Contact does wish to receive e-mail promotions from AdventureWorks, 2 = Contact does wish to receive e-mail promotions from AdventureWorks and selected partners. <b>Default:</b> 0
	AdditionalContactInfo	xml	Additional contact information about the person stored in xml format. <b>Nullable</b>
	Demographics	xml	Personal information such as hobbies, and income collected from online shoppers. Used for sales analysis. <b>Nullable</b>
PK	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

	Table	Join	Title / Name / Description
→	Person.BusinessEntity	<b>Person.Person</b> BusinessEntityID = Person.BusinessEntityBusinessEntityID	FK_Person_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID.

### Linked from

	Table	Join	Title / Name / Description
←	Person.BusinessEntityContact	<b>Person.Person</b> BusinessEntityID = Person.BusinessEntityContactPersonID	FK_BusinessEntityContact_Person_PersonID Foreign key constraint referencing Person.BusinessEntityID.
←	Sales.Customer	<b>Person.Person</b> BusinessEntityID = Sales.CustomerPersonID	FK_Customer_Person_PersonID Foreign key constraint referencing Person.BusinessEntityID.

Table		Join	Title / Name / Description
→ Person.EmailAddress		<b>Person.PersonBusinessEntityID = Person.EmailAddressBusinessEntityID</b>	FK_EmailAddress_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.
→ HumanResources.Employee		<b>Person.PersonBusinessEntityID = HumanResources.EmployeeBusinessEntityID</b>	FK_Employee_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.
→ Person.Password		<b>Person.PersonBusinessEntityID = Person.PasswordBusinessEntityID</b>	FK_Password_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.
→ Sales.PersonCreditCard		<b>Person.PersonBusinessEntityID = Sales.PersonCreditCardBusinessEntityID</b>	FK_PersonCreditCard_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.
→ Person.PersonPhone		<b>Person.PersonBusinessEntityID = Person.PersonPhoneBusinessEntityID</b>	FK_PersonPhone_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.

## Unique keys

Columns		Name / Description
key	BusinessEntityID	PK_Person_BusinessEntityID Primary key (clustered) constraint
key	rowguid	AK_Person_rowguid Unique nonclustered index. Used to support replication samples.

## Triggers

	Name	When	Description
⚡	iuPerson	After Insert, Update	AFTER INSERT, UPDATE trigger inserting Individual only if the Customer does not exist in the Store table and setting the ModifiedDate column in the Person table to the current date.

```

CREATE TRIGGER [Person].[iuPerson] ON [Person].[Person]
AFTER INSERT, UPDATE NOT FOR REPLICATION AS
BEGIN
    DECLARE @Count int;
    SET @Count = @@ROWCOUNT;
    IF @Count = 0
        RETURN;
    SET NOCOUNT ON;
    IF UPDATE ([BusinessEntityID]) OR UPDATE ([Demographics])
    BEGIN
        UPDATE [Person].[Person]
        SET [Person].[Person].[Demographics] = N'<IndividualSurvey
xmlns="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey">
<TotalPurchaseYTD>0.00</TotalPurchaseYTD>
</IndividualSurvey>'
        FROM inserted
        WHERE [Person].[Person].[BusinessEntityID] = inserted.[BusinessEntityID]
        AND inserted.[Demographics] IS NULL;
        UPDATE [Person].[Person]
        SET [Demographics].modify(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
insert <TotalPurchaseYTD>0.00</TotalPurchaseYTD>
as first
into (/IndividualSurvey)[1]')
        FROM inserted
        WHERE [Person].[Person].[BusinessEntityID] = inserted.[BusinessEntityID]
        AND inserted.[Demographics] IS NOT NULL
        AND inserted.[Demographics].exist(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
/IndividualSurvey/TotalPurchaseYTD') > 1;
    END;
END;

```

## Uses

	Name
grid	Person.Person
→	Person.BusinessEntity

Name

⚡ Person.iuPerson

grid Person.Person

Used By

Name

grid Person.Person

grid HumanResources.vEmployee

grid HumanResources.vEmployeeDepartment

grid HumanResources.vEmployeeDepartmentHistory

grid Person.vAdditionalContactInfo

grid Purchasing.vVendorWithContacts

grid Sales.vIndividualCustomer

grid Sales.vPersonDemographics

grid Sales.vSalesPerson

grid Sales.vSalesPersonSalesByFiscalYears

grid Sales.vStoreWithContacts

gear dbo.uspGetEmployeeManagers

gear dbo.uspGetManagerEmployees

fx dbo.ufnGetContactInformation

⚡ Person.iuPerson

⚡ Sales.iduSalesOrderDetail

→ HumanResources.Employee

→ Person.BusinessEntityContact

→ Person.EmailAddress

→ Person.Password

→ Person.PersonPhone

→ Sales.Customer

→ Sales.PersonCreditCard

## 8.2.1.11. Table: Person.PersonPhone

**Sample field: Status:** Active

Telephone number and type of a person.

### Columns

		Name	Data type	Description / Attributes
█	🔑	BusinessEntityID	int	Business entity identification number. Foreign key to Person.BusinessEntityID. <b>References:</b> Person.Person
█	🔑	PhoneNumber	Phone: nvarchar(25)	Telephone number identification number.
█	🔑	PhoneNumberTypeID	int	Kind of phone number. Foreign key to PhoneNumberType.PhoneNumberTypeID. <b>References:</b> Person.PhoneNumberType
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.Person	<b>Person.PersonPhoneBusinessEntityID</b> = Person.PersonBusinessEntityID	FK_PersonPhone_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.
→	Person.PhoneNumberType	<b>Person.PersonPhonePhoneNumberTypeID</b> = Person.PhoneNumberTypePhoneNum berTypeID	FK_PersonPhone_PhoneNumberType_PhoneNumberTypeID Foreign key constraint referencing PhoneNumberType.PhoneNumberTypeID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID, PhoneNumber, PhoneNumberTypeID	PK_PersonPhone_BusinessEntityID_PhoneNumber_PhoneNumberTypeID Primary key (clustered) constraint

### Uses

Name	
█	Person.PersonPhone
→	Person.Person
→	Person.PhoneNumberType

### Used By

Name	
█	Person.PersonPhone
↳	HumanResources.vEmployee
↳	Purchasing.vVendorWithContacts
↳	Sales.vIndividualCustomer
↳	Sales.vSalesPerson
↳	Sales.vStoreWithContacts

## 8.2.1.12. Table: Person.PhoneNumberType

**Sample field: Status:** Active

Type of phone number of a person.

### Columns

Name		Data type	Description / Attributes
☰	🔑 PhoneNumberTypeID	int	Primary key for telephone number type records. <b>Identity / Auto increment</b>
☰	Name	Name: nvarchar(50)	Name of the telephone number type
☰	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
→ Person.PersonPhone		Person.PhoneNumberType.PhoneNumberTypeID = Person.PersonPhone.PhoneNumberTypeID	FK_PersonPhone_PhoneNumberType_PhoneNumberTypeID Foreign key constraint referencing PhoneNumberType.PhoneNumberTypeID.

### Unique keys

Columns		Name / Description
🔑	PhoneNumberTypeID	PK_PhoneNumberType_PhoneNumberTypeID Primary key (clustered) constraint

### Used By

Name	
☰	Person.PhoneNumberType
↳	HumanResources.vEmployee
↳	Purchasing.vVendorWithContacts
↳	Sales.vIndividualCustomer
↳	Sales.vSalesPerson
↳	Sales.vStoreWithContacts
→	Person.PersonPhone

## 8.2.1.13. Table: Person.StateProvince

**Sample field: Status:** Active

State and province lookup table.

### Columns

		Name	Data type	Description / Attributes
		StateProvinceID	int	Primary key for StateProvince records. <b>Identity / Auto increment</b>
		StateProvinceCode	nchar(3)	ISO standard state or province code.
		CountryRegionCode	nvarchar(3)	ISO standard country or region code. Foreign key to CountryRegion.CountryRegionCode. <b>References:</b> Person.CountryRegion
		IsOnlyStateProvinceFlag	Flag: bit	0 = StateProvinceCode exists. 1 = StateProvinceCode unavailable, using CountryRegionCode. <b>Default:</b> 1
		Name	Name: nvarchar(50)	State or province description.
		TerritoryID	int	ID of the territory in which the state or province is located. Foreign key to SalesTerritory.SalesTerritoryID. <b>References:</b> Sales.SalesTerritory
		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
	Person.CountryRegion	Person.StateProvinceCountryRegionCode = Person.CountryRegionCountryRegionCode	FK_StateProvince_CountryRegion_CountryRegionCode Foreign key constraint referencing CountryRegion.CountryRegionCode.
	Sales.SalesTerritory	Person.StateProvinceTerritoryID = Sales.SalesTerritoryTerritoryID	FK_StateProvince_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.

### Linked from

Table		Join	Title / Name / Description
	Person.Address	Person.StateProvinceStateProvinceID = Person.AddressStateProvinceID	FK_Address_StateProvince_StateProvinceID Foreign key constraint referencing StateProvince.StateProvinceID.
	Sales.SalesTaxRate	Person.StateProvinceStateProvinceID = Sales.SalesTaxRateStateProvinceID	FK_SalesTaxRate_StateProvince_StateProvinceID Foreign key constraint referencing StateProvince.StateProvinceID.

### Unique keys

Columns		Name / Description
	StateProvinceID	PK_StateProvince_StateProvinceID Primary key (clustered) constraint
	Name	AK_StateProvince_Name Unique nonclustered index.
	rowguid	AK_StateProvince_rowguid Unique nonclustered index. Used to support replication samples.

Columns	Name / Description
 StateProvinceCode, CountryRegionCode	AK_StateProvince_StateProvinceCode_CountryRegionCode Unique nonclustered index.

## Uses

Name
 Person.StateProvince
→ Person.CountryRegion
→ Sales.SalesTerritory

## Used By

Name
 Person.StateProvince
↳ HumanResources.vEmployee
↳ Person.vStateProvinceCountryRegion
↳ Purchasing.vVendorWithAddresses
↳ Sales.vIndividualCustomer
↳ Sales.vSalesPerson
↳ Sales.vStoreWithAddresses
← Person.Address
← Sales.SalesTaxRate

## 8.2.2. Views

### 8.2.2.1. View: Person.vAdditionalContactInfo

**Sample field: Status:** Active

Displays the contact name and content from each element in the xml column AdditionalContactInfo for that person.

#### Columns

	Name	Data type	Description / Attributes
■	BusinessEntityID	int	
■	FirstName	Name: nvarchar(50)	
■	MiddleName	Name: nvarchar(50)	<b>Nullable</b>
■	LastName	Name: nvarchar(50)	
■	TelephoneNumber	nvarchar(50)	<b>Nullable</b>
■	TelephoneSpecialInstructions	nvarchar(MAX)	<b>Nullable</b>
■	Street	nvarchar(50)	<b>Nullable</b>
■	City	nvarchar(50)	<b>Nullable</b>
■	StateProvince	nvarchar(50)	<b>Nullable</b>
■	PostalCode	nvarchar(50)	<b>Nullable</b>
■	CountryRegion	nvarchar(50)	<b>Nullable</b>
■	HomeAddressSpecialInstructions	nvarchar(MAX)	<b>Nullable</b>
■	EMailAddress	nvarchar(128)	<b>Nullable</b>
■	EMailSpecialInstructions	nvarchar(MAX)	<b>Nullable</b>
■	EMailTelephoneNumber	nvarchar(50)	<b>Nullable</b>
■	rowguid	uniqueidentifier	
■	ModifiedDate	datetime	

#### Uses

	Name
■	Person.vAdditionalContactInfo
■	Person.Person
?	ContactInfo.ref.value

## Script

```
CREATE VIEW [Person].[vAdditionalContactInfo]
AS
SELECT
    [BusinessEntityID]
    ,[FirstName]
    ,[MiddleName]
    ,[LastName]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo";
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:telephoneNumber)[1]/act:number', 'nvarchar(50)' ) AS [TelephoneNumber]
    ,LTRIM(RTRIM([ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo";
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:telephoneNumber/act:SpecialInstructions/text())[1]', 'nvarchar(max)')))) AS [TelephoneSpecialInstructions]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:homePostalAddress/act:Street)[1]', 'nvarchar(50)' ) AS [Street]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:homePostalAddress/act:City)[1]', 'nvarchar(50)' ) AS [City]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:homePostalAddress/act:StateProvince)[1]', 'nvarchar(50)' ) AS [StateProvince]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:homePostalAddress/act:PostalCode)[1]', 'nvarchar(50)' ) AS [PostalCode]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:homePostalAddress/act:CountryRegion)[1]', 'nvarchar(50)' ) AS [CountryRegion]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:homePostalAddress/act:SpecialInstructions/text())[1]', 'nvarchar(max)' ) AS [HomeAddressSpecialInstructions]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:eMail/act:eEmailAddress)[1]', 'nvarchar(128)' ) AS [EEmailAddress]
    ,LTRIM(RTRIM([ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:eMail/act:SpecialInstructions/text())[1]', 'nvarchar(max)')))) AS [EMailSpecialInstructions]
    ,[ContactInfo].ref.value(N'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ContactInfo';
    declare namespace act="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactTypes";
    (act:eMail/act:SpecialInstructions/act:telephoneNumber/act:number)[1]', 'nvarchar(50)' ) AS [EMailTelephoneNumber]
    ,[rowguid]
    ,[ModifiedDate]
FROM [Person].[Person]
OUTER APPLY [AdditionalContactInfo].nodes(
    'declare namespace ci="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ContactInfo";
    /ci:AdditionalContactInfo') AS ContactInfo(ref)
WHERE [AdditionalContactInfo] IS NOT NULL;
```

## 8.2.2.2. View: Person.vStateProvinceCountryRegion

**Sample field: Status:** Active

Joins StateProvince table with CountryRegion table.

### Columns

Name		Data type	Description / Attributes
	StateProvinceID	int	
	StateProvinceCode	nchar(3)	
	IsOnlyStateProvinceFlag	Flag: bit	
	StateProvinceName	Name: nvarchar(50)	
	TerritoryID	int	
	CountryRegionCode	nvarchar(3)	
	CountryRegionName	Name: nvarchar(50)	

### Unique keys

Columns		Name / Description
	StateProvinceID, CountryRegionCode	IX_vStateProvinceCountryRegion Clustered index on the view vStateProvinceCountryRegion.

### Uses

Name	
	Person.vStateProvinceCountryRegion
	Person.CountryRegion
	Person.StateProvince

### Script

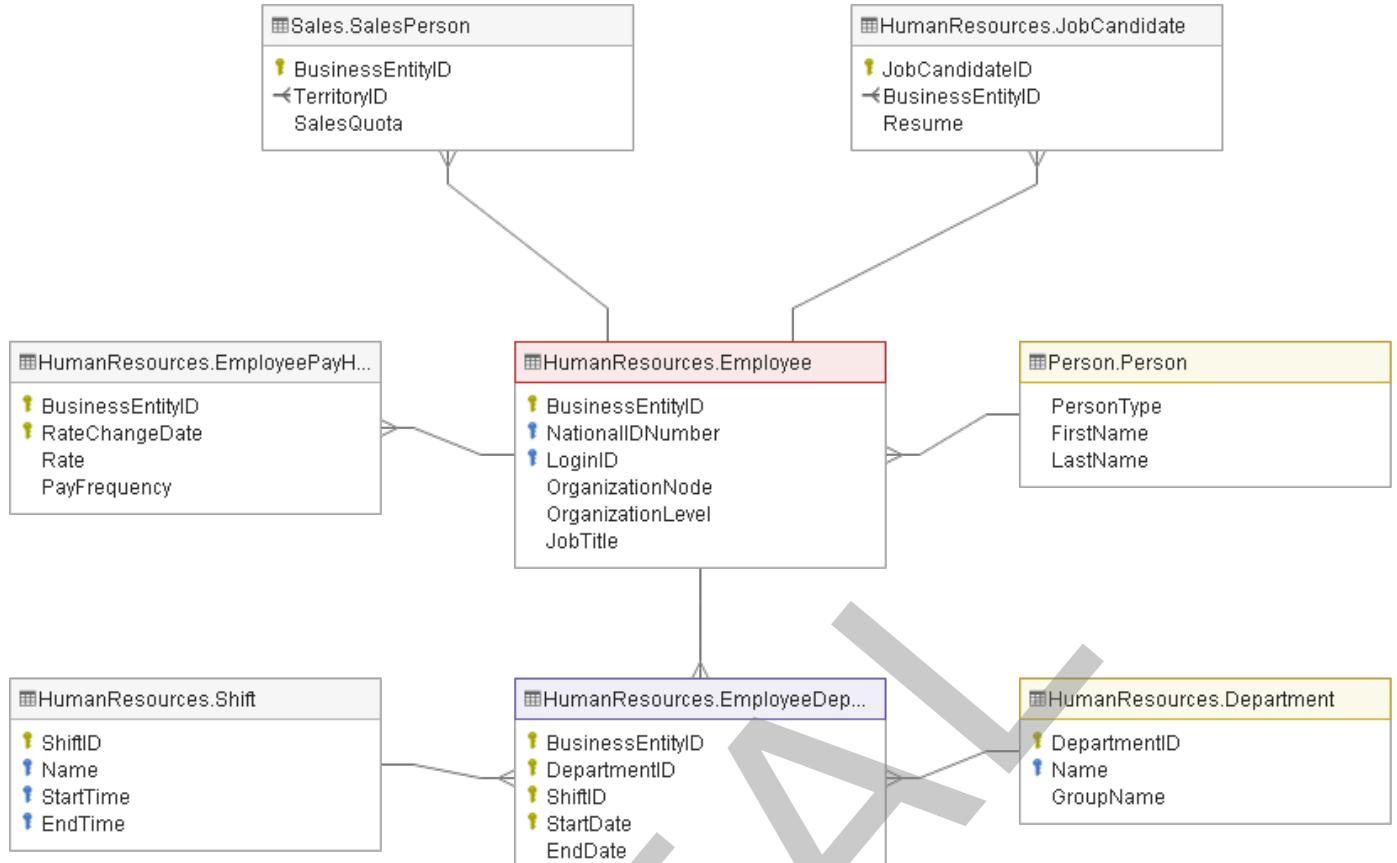
```

CREATE VIEW [Person].[vStateProvinceCountryRegion]
WITH SCHEMABINDING
AS
SELECT
    sp.[StateProvinceID]
    ,sp.[StateProvinceCode]
    ,sp.[IsOnlyStateProvinceFlag]
    ,sp.[Name] AS [StateProvinceName]
    ,sp.[TerritoryID]
    ,cr.[CountryRegionCode]
    ,cr.[Name] AS [CountryRegionName]
FROM [Person].[StateProvince] sp
INNER JOIN [Person].[CountryRegion] cr
ON sp.[CountryRegionCode] = cr.[CountryRegionCode];

```

TRIAL

### 8.3. Human Resources



Employees of Adventure Works Cycles.

## 8.3.1. Tables

### 8.3.1.1. Table: HumanResources.Department

**Sample field: Status:** Active

Lookup table containing the departments within the Adventure Works Cycles company.

#### Columns

		Name	Data type	Description / Attributes
		DepartmentID	smallint	Primary key for Department records. <b>Identity / Auto increment</b>
		Name	Name: nvarchar(50)	Name of the department.
		GroupName	Name: nvarchar(50)	Name of the group to which the department belongs.
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

Table		Join	Title / Name / Description
	HumanResources.EmployeeDepartmentHistory	<b>HumanResources.Department</b> DepartmentID = HumanResources.EmployeeDepartmentHistoryDepartmentID	FK_EmployeeDepartmentHistory_Department_DepartmentID Foreign key constraint referencing Department.DepartmentID.

#### Unique keys

Columns		Name / Description
	DepartmentID	PK_Department_DepartmentID Primary key (clustered) constraint
	Name	AK_Department_Name Unique nonclustered index.

#### Used By

		Name
	HumanResources.Department	
	HumanResources.vEmployeeDepartment	
	HumanResources.vEmployeeDepartmentHistory	
	HumanResources.EmployeeDepartmentHistory	

### 8.3.1.2. Table: HumanResources.Employee

**Sample field: Status:** Active

Employee information such as salary, department, and title.

#### Columns

		Name	Data type	Description / Attributes
■	🔑	BusinessEntityID	int	Primary key for Employee records. Foreign key to BusinessEntity.BusinessEntityID. <b>References:</b> Person.Person
■	🔑	NationalIDNumber	nvarchar(15)	Unique national identification number such as a social security number.
■	🔑	LoginID	nvarchar(256)	Network login. Test2
■		OrganizationNode	hierarchyid	Where the employee is located in corporate hierarchy. <b>Nullable</b>
■		OrganizationLevel	smallint	The depth of the employee in the corporate hierarchy. <b>Nullable</b> <b>Computed:</b> ([OrganizationNode].[GetLevel]())
■		JobTitle	nvarchar(50)	Work title such as Buyer or Sales Representative.
■		BirthDate	date	Date of birth.
■		MaritalStatus	nchar(1)	M = Married, S = Single
■		Gender	nchar(1)	M = Male, F = Female
■		HireDate	date	Employee hired on this date.
■		SalariedFlag	Flag: bit	Job classification. 0 = Hourly, not exempt from collective bargaining. 1 = Salaried, exempt from collective bargaining. <b>Default:</b> 1
■		VacationHours	smallint	Number of available vacation hours. <b>Default:</b> 0
■		SickLeaveHours	smallint	Number of available sick leave hours. <b>Default:</b> 0
■		CurrentFlag	Flag: bit	0 = Inactive, 1 = Active <b>Default:</b> 1
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
→	Person.Person	<b>HumanResources.Employee</b> BusinessEntityID = Person.PersonBusinessEntityID	FK_Employee_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.

#### Linked from

Table		Join	Title / Name / Description
→	HumanResources.EmployeeDepartmentHistory	<b>HumanResources.Employee</b> BusinessEntityID = HumanResources.EmployeeDepartmentHistoryBusinessEntityID	FK_EmployeeDepartmentHistory_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.

	Table	Join	Title / Name / Description
→	HumanResources.EmployeePayHistory	<b>HumanResources.EmployeeBusinessEntityID =</b> HumanResources.EmployeePayHistory.BusinessEntityID	FK_EmployeePayHistory_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.
→	HumanResources.JobCandidate	<b>HumanResources.EmployeeBusinessEntityID =</b> HumanResources.JobCandidateBusinessEntityID	FK_JobCandidate_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.
→	Purchasing.PurchaseOrderHeader	<b>HumanResources.EmployeeBusinessEntityID =</b> Purchasing.PurchaseOrderHeaderEmployeeID	FK_PurchaseOrderHeader_Employee_EmployeeID Foreign key constraint referencing Employee.EmployeeID.
→	Sales.SalesPerson	<b>HumanResources.EmployeeBusinessEntityID =</b> Sales.SalesPersonBusinessEntityID	FK_SalesPerson_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.

## Unique keys

	Columns	Name / Description
🔑	BusinessEntityID	PK_Employee_BusinessEntityID Primary key (clustered) constraint
🔑	LoginID	AK_Employee_LoginID Unique nonclustered index.
🔑	NationalIDNumber	AK_Employee_NationalIDNumber Unique nonclustered index.
🔑	rowguid	AK_Employee_rowguid Unique nonclustered index. Used to support replication samples.

## Triggers

	Name	When	Description
⚡	dEmployee	Instead Of Delete	INSTEAD OF DELETE trigger which keeps Employees from being deleted.
<pre>CREATE TRIGGER [HumanResources].[dEmployee] ON HumanResources.Employee INSTEAD OF DELETE NOT FOR REPLICATION AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN         RAISERROR             (N'Employees cannot be deleted. They can only be marked as not current.', -- Message             10, -- Severity.             1); -- State.         -- Rollback any active or uncommittable transactions         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END     END; END;</pre>			

## Uses

	Name
grid	HumanResources.Employee
grid	HumanResources.Employee
→	Person.Person

## Used By

Name
HumanResources.Employee
HumanResources.Employee
HumanResources.vEmployee
HumanResources.vEmployeeDepartment
HumanResources.vEmployeeDepartmentHistory
Sales.vSalesPerson
Sales.vSalesPersonSalesByFiscalYears
dbo.uspGetEmployeeManagers
dbo.uspGetManagerEmployees
HumanResources.uspUpdateEmployeeHireInfo
HumanResources.uspUpdateEmployeeLogin
HumanResources.uspUpdateEmployeePersonalInfo
dbo.ufnGetContactInformation
→ HumanResources.EmployeeDepartmentHistory
→ HumanResources.EmployeePayHistory
→ HumanResources.JobCandidate
→ Purchasing.PurchaseOrderHeader
→ Sales.SalesPerson

### 8.3.1.3. Table: HumanResources.EmployeeDepartmentHistory

**Sample field: Status:** Active

Employee department transfers.

#### Columns

Name			Data type	Description / Attributes
BusinessEntityID	Business Entity ID	int		Employee identification number. Foreign key to Employee.BusinessEntityID. <b>References:</b> HumanResources.Employee
DepartmentID	Department ID	smallint		Department in which the employee worked including currently. Foreign key to Department.DepartmentID. <b>References:</b> HumanResources.Department
ShiftID	Shift ID	tinyint		Identifies which 8-hour shift the employee works. Foreign key to Shift.Shift.ID. <b>References:</b> HumanResources.Shift
StartDate	Start Date	date		Date the employee started work in the department.
EndDate	End Date	date		Date the employee left the department. NULL = Current department. <b>Nullable</b>
ModifiedDate	Modified Date	datetime		Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table	Join	Title / Name / Description
HumanResources.Department	HumanResources.EmployeeDepartmentHistory DepartmentID = HumanResources.DepartmentDepartmentID	FK_EmployeeDepartmentHistory_Department_DepartmentID Foreign key constraint referencing Department.DepartmentID.
HumanResources.Employee	HumanResources.EmployeeDepartmentHistory BusinessEntityID = HumanResources.EmployeeBusinessEntityID	FK_EmployeeDepartmentHistory_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.
HumanResources.Shift	HumanResources.EmployeeDepartmentHistory ShiftID = HumanResources.ShiftShiftID	FK_EmployeeDepartmentHistory_Shift_ShiftID Foreign key constraint referencing Shift.ShiftID

#### Unique keys

Columns	Name / Description
BusinessEntityID, StartDate, DepartmentID, ShiftID	PK_EmployeeDepartmentHistory_BusinessEntityID_StartDate_DepartmentID Primary key (clustered) constraint

#### Uses

Name
HumanResources.EmployeeDepartmentHistory
→ HumanResources.Department
→ HumanResources.Employee
→ HumanResources.Shift

#### Used By

Name
HumanResources.EmployeeDepartmentHistory

Name

HumanResources.vEmployeeDepartment

HumanResources.vEmployeeDepartmentHistory

TRIAL

### 8.3.1.4. Table: HumanResources.EmployeePayHistory

**Sample field: Status:** Active

Employee pay history.

#### Columns

		Name	Data type	Description / Attributes
■	🔑	BusinessEntityID	int	Employee identification number. Foreign key to Employee.BusinessEntityID. <b>References:</b> HumanResources.Employee
■	🔑	RateChangeDate	datetime	Date the change in pay is effective
■		Rate	money	Salary hourly rate.
■		PayFrequency	tinyint	1 = Salary received monthly, 2 = Salary received biweekly
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
→	HumanResources.Employee	HumanResources.EmployeePayHistory BusinessEntityID = HumanResources.EmployeeBusinessE ntityID	FK_EmployeePayHistory_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.

#### Unique keys

Columns		Name / Description
🔑	BusinessEntityID, RateChangeDate	PK_EmployeePayHistory_BusinessEntityID_RateChangeDate Primary key (clustered) constraint

#### Uses

Name	
█	HumanResources.EmployeePayHistory
→	HumanResources.Employee

#### Used By

Name	
█	HumanResources.EmployeePayHistory
⚙️	HumanResources.uspUpdateEmployeeHireInfo

### 8.3.1.5. Table: HumanResources.JobCandidate

**Sample field: Status:** Active

Résumés submitted to Human Resources by job applicants.

#### Columns

		Name	Data type	Description / Attributes
>List	JobCandidateID	Primary key for JobCandidate records. <b>Identity / Auto increment</b>		
List	BusinessEntityID	Employee identification number if applicant was hired. Foreign key to Employee.BusinessEntityID. <b>Nullable</b> <b>References:</b> HumanResources.Employee		
List	Resume	Résumé in XML format. <b>Nullable</b>		
List	ModifiedDate	Date and time the record was last updated. <b>Default:</b> getdate()		

#### Links to

Table		Join	Title / Name / Description
→ HumanResources.Employee		HumanResources.JobCandidateBusinessEntityID = HumanResources.EmployeeBusinessEntityID	FK_JobCandidate_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.

#### Unique keys

Columns		Name / Description
JobCandidateID		PK_JobCandidate_JobCandidateID Primary key (clustered) constraint

#### Uses

Name	
HumanResources.JobCandidate	
→ HumanResources.Employee	

#### Used By

Name	
HumanResources.JobCandidate	
↳ HumanResources.vJobCandidate	
↳ HumanResources.vJobCandidateEducation	
↳ HumanResources.vJobCandidateEmployment	
⚙️ dbo.uspSearchCandidateResumes	

### 8.3.1.6. Table: HumanResources.Shift

**Sample field: Status:** Active

Work shift lookup table.

#### Columns

		Name	Data type	Description / Attributes
grid	key	ShiftID	tinyint	Primary key for Shift records. <b>Identity / Auto increment</b>
grid	key	Name	Name: nvarchar(50)	Shift description.
grid	key	StartTime	time(7)	Shift start time.
grid	key	EndTime	time(7)	Shift end time.
grid		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

Table		Join	Title / Name / Description
→	HumanResources.EmployeeDepartmentHistory	<b>HumanResources.Shift</b> ShiftID = HumanResources.EmployeeDepartmentHistoryShiftID	FK_EmployeeDepartmentHistory_Shift_ShiftID Foreign key constraint referencing Shift.ShiftID

#### Unique keys

Columns		Name / Description
key	ShiftID	PK_Shift_ShiftID Primary key (clustered) constraint
key	Name	AK_Shift_Name Unique nonclustered index.
key	StartTime, EndTime	AK_Shift_StartTime_EndTime Unique nonclustered index.

#### Used By

		Name
grid	HumanResources.Shift	
grid	HumanResources.vEmployeeDepartmentHistory	
→	HumanResources.EmployeeDepartmentHistory	

## 8.3.2. Views

### 8.3.2.1. View: HumanResources.vEmployee

**Sample field: Status:** Active

Employee names and addresses.

#### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Title	nvarchar(8)	<b>Nullable</b>
3	FirstName	Name: nvarchar(50)	
4	MiddleName	Name: nvarchar(50)	<b>Nullable</b>
5	LastName	Name: nvarchar(50)	
6	Suffix	nvarchar(10)	<b>Nullable</b>
7	JobTitle	nvarchar(50)	
8	PhoneNumber	Phone: nvarchar(25)	<b>Nullable</b>
9	PhoneNumberType	Name: nvarchar(50)	<b>Nullable</b>
10	EmailAddress	nvarchar(50)	<b>Nullable</b>
11	EmailPromotion	int	
12	AddressLine1	nvarchar(60)	
13	AddressLine2	nvarchar(60)	<b>Nullable</b>
14	City	nvarchar(30)	
15	StateProvinceName	Name: nvarchar(50)	
16	PostalCode	nvarchar(15)	
17	CountryRegionName	Name: nvarchar(50)	
18	AdditionalContactInfo	xml	<b>Nullable</b>

#### Uses

	Name
1	HumanResources.vEmployee
2	HumanResources.Employee
3	Person.Address
4	Person.BusinessEntityAddress
5	Person.CountryRegion
6	Person.EmailAddress
7	Person.Person
8	Person.PersonPhone
9	Person.PhoneNumberType
10	Person.StateProvince

## Script

```
CREATE VIEW [HumanResources].[vEmployee]
AS
SELECT
    e.[BusinessEntityID]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,e.[JobTitle]
    ,pp.[PhoneNumber]
    ,pnt.[Name] AS [PhoneNumberType]
    ,ea.[EmailAddress]
    ,p.[EmailPromotion]
    ,a.[AddressLine1]
    ,a.[AddressLine2]
    ,a.[City]
    ,sp.[Name] AS [StateProvinceName]
    ,a.[PostalCode]
    ,cr.[Name] AS [CountryRegionName]
    ,p.[AdditionalContactInfo]
FROM [HumanResources].[Employee] e
    INNER JOIN [Person].[Person] p
        ON p.[BusinessEntityID] = e.[BusinessEntityID]
    INNER JOIN [Person].[BusinessEntityAddress] bea
        ON bea.[BusinessEntityID] = e.[BusinessEntityID]
    INNER JOIN [Person].[Address] a
        ON a.[AddressID] = bea.[AddressID]
    INNER JOIN [Person].[StateProvince] sp
        ON sp.[StateProvinceID] = a.[StateProvinceID]
    INNER JOIN [Person].[CountryRegion] cr
        ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
    LEFT OUTER JOIN [Person].[PersonPhone] pp
        ON pp.BusinessEntityID = p.[BusinessEntityID]
    LEFT OUTER JOIN [Person].[PhoneNumberType] pnt
        ON pp.[PhoneNumberTypeID] = pnt.[PhoneNumberTypeID]
    LEFT OUTER JOIN [Person].[EmailAddress] ea
        ON p.[BusinessEntityID] = ea.[BusinessEntityID];
```

TRIAL

### 8.3.2.2. View: HumanResources.vEmployeeDepartment

#### Sample field: Status: Active

Returns employee name, title, and current department.

#### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Title	nvarchar(8)	Nullable
3	FirstName	Name: nvarchar(50)	
4	MiddleName	Name: nvarchar(50)	Nullable
5	LastName	Name: nvarchar(50)	
6	Suffix	nvarchar(10)	Nullable
7	JobTitle	nvarchar(50)	
8	Department	Name: nvarchar(50)	
9	GroupName	Name: nvarchar(50)	
10	StartDate	date	

#### Uses

Name
HumanResources.vEmployeeDepartment
HumanResources.Department
HumanResources.Employee
HumanResources.EmployeeDepartmentHistory
Person.Person

#### Script

```
CREATE VIEW [HumanResources].[vEmployeeDepartment]
AS
SELECT
    e.[BusinessEntityID]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,e.[JobTitle]
    ,d.[Name] AS [Department]
    ,d.[GroupName]
    ,edh.[StartDate]
FROM [HumanResources].[Employee] e
    INNER JOIN [Person].[Person] p
        ON p.[BusinessEntityID] = e.[BusinessEntityID]
    INNER JOIN [HumanResources].[EmployeeDepartmentHistory] edh
        ON e.[BusinessEntityID] = edh.[BusinessEntityID]
    INNER JOIN [HumanResources].[Department] d
        ON edh.[DepartmentID] = d.[DepartmentID]
WHERE edh.EndDate IS NULL
```

### 8.3.2.3. View: HumanResources.vEmployeeDepartmentHistory

#### Sample field: Status: Active

Returns employee name and current and previous departments.

#### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Title	nvarchar(8)	Nullable
3	FirstName	Name: nvarchar(50)	
4	MiddleName	Name: nvarchar(50)	Nullable
5	LastName	Name: nvarchar(50)	
6	Suffix	nvarchar(10)	Nullable
7	Shift	Name: nvarchar(50)	
8	Department	Name: nvarchar(50)	
9	GroupName	Name: nvarchar(50)	
10	StartDate	date	
11	EndDate	date	Nullable

#### Uses

	Name
1	HumanResources.vEmployeeDepartmentHistory
2	HumanResources.Department
3	HumanResources.Employee
4	HumanResources.EmployeeDepartmentHistory
5	HumanResources.Shift
6	Person.Person

#### Script

```

CREATE VIEW [HumanResources].[vEmployeeDepartmentHistory]
AS
SELECT
    e.[BusinessEntityID]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,s.[Name] AS [Shift]
    ,d.[Name] AS [Department]
    ,d.[GroupName]
    ,edh.[StartDate]
    ,edh.[EndDate]
FROM [HumanResources].[Employee] e
    INNER JOIN [Person].[Person] p
        ON p.[BusinessEntityID] = e.[BusinessEntityID]
    INNER JOIN [HumanResources].[EmployeeDepartmentHistory] edh
        ON e.[BusinessEntityID] = edh.[BusinessEntityID]
    INNER JOIN [HumanResources].[Department] d
        ON edh.[DepartmentID] = d.[DepartmentID]
    INNER JOIN [HumanResources].[Shift] s
        ON s.[ShiftID] = edh.[ShiftID];

```

### 8.3.2.4. View: HumanResources.vJobCandidate

#### Sample field: Status: Active

Job candidate names and resumes.

#### Columns

	Name	Data type	Description / Attributes
JobCandidateID	int	Identity / Auto increment	
BusinessEntityID	int	Nullable	
Name.Prefix	nvarchar(30)	Nullable	
Name.First	nvarchar(30)	Nullable	
Name.Middle	nvarchar(30)	Nullable	
Name.Last	nvarchar(30)	Nullable	
Name.Suffix	nvarchar(30)	Nullable	
Skills	nvarchar(MAX)	Nullable	
Addr.Type	nvarchar(30)	Nullable	
Addr.Loc.CountryRegion	nvarchar(100)	Nullable	
Addr.Loc.State	nvarchar(100)	Nullable	
Addr.Loc.City	nvarchar(100)	Nullable	
Addr.PostalCode	nvarchar(20)	Nullable	
EMail	nvarchar(MAX)	Nullable	
WebSite	nvarchar(MAX)	Nullable	
ModifiedDate	datetime		

#### Uses

Name
HumanResources.vJobCandidate
HumanResources.JobCandidate
Resume.ref.value

## Script

```
CREATE VIEW [HumanResources].[vJobCandidate]
AS
SELECT
    jc.[JobCandidateID]
    ,jc.[BusinessEntityID]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/Name/Name.Prefix)[1]', 'nvarchar(30)' ) AS [Name.Prefix]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/Name/Name.First)[1]', 'nvarchar(30)' ) AS [Name.First]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/Name/Name.Middle)[1]', 'nvarchar(30)' ) AS [Name.Middle]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/Name/Name.Last)[1]', 'nvarchar(30)' ) AS [Name.Last]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/Name/Name.Suffix)[1]', 'nvarchar(30)' ) AS [Name.Suffix]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/Skills)[1]', 'nvarchar(max)' ) AS [Skills]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Address/Addr.Type)[1]', 'nvarchar(30)' ) AS [Addr.Type]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Address/Addr.Location/Location/Loc.CountryRegion)[1]', 'nvarchar(100)' ) AS [Addr.Loc.CountryRegion]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Address/Addr.Location/Location/Loc.State)[1]', 'nvarchar(100)' ) AS [Addr.Loc.State]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Address/Addr.Location/Location/Loc.City)[1]', 'nvarchar(100)' ) AS [Addr.Loc.City]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Address/Addr.PostalCode)[1]', 'nvarchar(20)' ) AS [Addr.PostalCode]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/EMail)[1]', 'nvarchar(max)' ) AS [EMail]
    ,[Resume].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,'/Resume/WebSite)[1]', 'nvarchar(max)' ) AS [WebSite]
    ,jc.[ModifiedDate]
FROM [HumanResources].[JobCandidate] jc
CROSS APPLY jc.[Resume].nodes(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
    ,'/Resume') AS Resume(ref);
```

### 8.3.2.5. View: HumanResources.vJobCandidateEducation

#### Sample field: Status: Active

Displays the content from each education related element in the xml column Resume in the HumanResources.JobCandidate table. The content has been localized into French, Simplified Chinese and Thai. Some data may not display correctly unless supplemental language support is installed.

#### Columns

	Name	Data type	Description / Attributes
1	JobCandidateID	int	Identity / Auto increment
2	Edu.Level	nvarchar(MAX)	Nullable
3	Edu.StartDate	datetime	Nullable
4	Edu.EndDate	datetime	Nullable
5	Edu.Degree	nvarchar(50)	Nullable
6	Edu.Major	nvarchar(50)	Nullable
7	Edu.Minor	nvarchar(50)	Nullable
8	Edu.GPA	nvarchar(5)	Nullable
9	Edu.GPAScale	nvarchar(5)	Nullable
10	Edu.School	nvarchar(100)	Nullable
11	Edu.Loc.CountryRegion	nvarchar(100)	Nullable
12	Edu.Loc.State	nvarchar(100)	Nullable
13	Edu.Loc.City	nvarchar(100)	Nullable

#### Uses

Name
HumanResources.vJobCandidateEducation
HumanResources.JobCandidate
Education.ref.value

## Script

```
CREATE VIEW [HumanResources].[vJobCandidateEducation]
AS
SELECT
    jc.[JobCandidateID]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Level)[1], 'nvarchar(max)') AS [Edu.Level]
    ,CONVERT(datetime, REPLACE([Education].ref.value(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
(Edu.StartDate)[1], 'nvarchar(20)', 'Z', ''), 101) AS [Edu.StartDate]
    ,CONVERT(datetime, REPLACE([Education].ref.value(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";
(Edu.EndDate)[1], 'nvarchar(20)', 'Z', ''), 101) AS [Edu.EndDate]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Degree)[1], 'nvarchar(50)') AS [Edu.Degree]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Major)[1], 'nvarchar(50)') AS [Edu.Major]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Minor)[1], 'nvarchar(50)') AS [Edu.Minor]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.GPA)[1], 'nvarchar(5)') AS [Edu.GPA]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.GPAScale)[1], 'nvarchar(5)') AS [Edu.GPAScale]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.School)[1], 'nvarchar(100)') AS [Edu.School]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Location/Location/Loc.CountryRegion)[1], 'nvarchar(100)') AS [Edu.Loc.CountryRegion]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Location/Location/Loc.State)[1], 'nvarchar(100)') AS [Edu.Loc.State]
    ,[Education].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
        ,(Edu.Location/Location/Loc.City)[1], 'nvarchar(100)') AS [Edu.Loc.City]
FROM [HumanResources].[JobCandidate] jc
CROSS APPLY jc.[Resume].nodes(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume";'
    /Resume/Education') AS [Education](ref);
```

### 8.3.2.6. View: HumanResources.vJobCandidateEmployment

#### Sample field: Status: Active

Displays the content from each employment history related element in the xml column Resume in the HumanResources.JobCandidate table. The content has been localized into French, Simplified Chinese and Thai. Some data may not display correctly unless supplemental language support is installed.

#### Columns

	Name	Data type	Description / Attributes
JobCandidateID	int		Identity / Auto increment
Emp.StartDate	datetime		Nullable
Emp.EndDate	datetime		Nullable
Emp.OrgName	nvarchar(100)		Nullable
Emp.JobTitle	nvarchar(100)		Nullable
Emp.Responsibility	nvarchar(MAX)		Nullable
Emp.FunctionCategory	nvarchar(MAX)		Nullable
Emp.IndustryCategory	nvarchar(MAX)		Nullable
Emp.Loc.CountryRegion	nvarchar(MAX)		Nullable
Emp.Loc.State	nvarchar(MAX)		Nullable
Emp.Loc.City	nvarchar(MAX)		Nullable

#### Uses

Name
HumanResources.vJobCandidateEmployment
HumanResources.JobCandidate
Employment.ref.value

## Script

```
CREATE VIEW [HumanResources].[vJobCandidateEmployment]
AS
SELECT
    jc.[JobCandidateID]
    ,CONVERT(datetime, REPLACE([Employment].ref.value(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume",
(Emp.StartDate)[1]', 'nvarchar(20)'), 'Z', ''), 101) AS [Emp.StartDate]
    ,CONVERT(datetime, REPLACE([Employment].ref.value(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume",
(Emp.EndDate)[1]', 'nvarchar(20)'), 'Z', ''), 101) AS [Emp.EndDate]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.OrgName)[1], 'nvarchar(100)' AS [Emp.OrgName]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.JobTitle)[1], 'nvarchar(100)' AS [Emp.JobTitle]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.Responsibility)[1], 'nvarchar(max)' AS [Emp.Responsibility]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.FunctionCategory)[1], 'nvarchar(max)' AS [Emp.FunctionCategory]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.IndustryCategory)[1], 'nvarchar(max)' AS [Emp.IndustryCategory]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.Location/Location/Loc.CountryRegion)[1], 'nvarchar(max)' AS [Emp.Loc.CountryRegion]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.Location/Location/Loc.State)[1], 'nvarchar(max)' AS [Emp.Loc.State]
    ,[Employment].ref.value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    (Emp.Location/Location/Loc.City)[1], 'nvarchar(max)' AS [Emp.Loc.City]
FROM [HumanResources].[JobCandidate] jc
CROSS APPLY jc.[Resume].nodes(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/Resume';
    /Resume/Employment') AS Employment(ref);
```

## 8.3.3. Procedures

### 8.3.3.1. Procedure: dbo.uspGetEmployeeManagers

Stored procedure using a recursive query to return the direct and indirect managers of the specified employee.

#### Input/Output

	Name	Data type	Description
@	BusinessEntityID	int	Input parameter for the stored procedure uspGetEmployeeManagers. Enter a valid BusinessEntityID from the HumanResources.Employee table.

#### Uses

	Name
dbo.uspGetEmployeeManagers	
HumanResources.Employee	
Person.Person	
EMP_cte.OrganizationNode.GetAncestor	
EMP_cte.OrganizationNode.ToString	

#### Script

```
CREATE PROCEDURE [dbo].[uspGetEmployeeManagers]
    @BusinessEntityID [int]
AS
BEGIN
    SET NOCOUNT ON;
    -- Use recursive query to list out all Employees required for a particular Manager
    WITH [EMP_cte]([BusinessEntityID], [OrganizationNode], [FirstName], [LastName], [JobTitle], [RecursionLevel]) -- CTE name
and columns
    AS (
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName], p.[LastName], e.[JobTitle], 0 -- Get the initial
Employee
        FROM [HumanResources].[Employee] e
            INNER JOIN [Person].[Person] as p
                ON p.[BusinessEntityID] = e.[BusinessEntityID]
        WHERE e.[BusinessEntityID] = @BusinessEntityID
        UNION ALL
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName], p.[LastName], e.[JobTitle], [RecursionLevel] + 1 -- --
Join recursive member to anchor
        FROM [HumanResources].[Employee] e
            INNER JOIN [EMP_cte]
                ON e.[OrganizationNode] = [EMP_cte].[OrganizationNode].GetAncestor(1)
            INNER JOIN [Person].[Person] p
                ON p.[BusinessEntityID] = e.[BusinessEntityID]
    )
    -- Join back to Employee to return the manager name
    SELECT [EMP_cte].[RecursionLevel], [EMP_cte].[BusinessEntityID], [EMP_cte].[FirstName], [EMP_cte].[LastName],
    [EMP_cte].[OrganizationNode].ToString() AS [OrganizationNode], p.[FirstName] AS 'ManagerFirstName', p.[LastName] AS
'ManagerLastName' -- Outer select from the CTE
    FROM [EMP_cte]
        INNER JOIN [HumanResources].[Employee] e
            ON [EMP_cte].[OrganizationNode].GetAncestor(1) = e.[OrganizationNode]
        INNER JOIN [Person].[Person] p
            ON p.[BusinessEntityID] = e.[BusinessEntityID]
    ORDER BY [RecursionLevel], [EMP_cte].[OrganizationNode].ToString()
    OPTION (MAXRECURSION 25)
END;
```

### 8.3.3.2. Procedure: dbo.uspGetManagerEmployees

Stored procedure using a recursive query to return the direct and indirect employees of the specified manager.

#### Input/Output

	Name	Data type	Description
→@	BusinessEntityID	int	Input parameter for the stored procedure uspGetManagerEmployees. Enter a valid BusinessEntityID of the manager from the HumanResources.Employee table.

#### Uses

	Name
⚙️	dbo.uspGetManagerEmployees
💻	HumanResources.Employee
💻	Person.Person
⚡?	e.OrganizationNode.GetAncestor
⚡?	EMP_cte.OrganizationNode.GetAncestor
⚡?	EMP_cte.OrganizationNode.ToString

#### Script

```

CREATE PROCEDURE [dbo].[uspGetManagerEmployees]
    @BusinessEntityID [int]
AS
BEGIN
    SET NOCOUNT ON;
    -- Use recursive query to list out all Employees required for a particular Manager
    WITH [EMP_cte]([BusinessEntityID], [OrganizationNode], [FirstName], [LastName], [RecursionLevel]) -- CTE name and columns
    AS (
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName], p.[LastName], 0 -- Get the initial list of
        Employees for Manager n
        FROM [HumanResources].[Employee] e
            INNER JOIN [Person].[Person] p
                ON p.[BusinessEntityID] = e.[BusinessEntityID]
        WHERE e.[BusinessEntityID] = @BusinessEntityID
        UNION ALL
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName], p.[LastName], [RecursionLevel] + 1 -- Join
        recursive member to anchor
        FROM [HumanResources].[Employee] e
            INNER JOIN [EMP_cte]
                ON e.[OrganizationNode].GetAncestor(1) = [EMP_cte].[OrganizationNode]
                    INNER JOIN [Person].[Person] p
                        ON p.[BusinessEntityID] = e.[BusinessEntityID]
    )
    -- Join back to Employee to return the manager name
    SELECT [EMP_cte].[RecursionLevel], [EMP_cte].[OrganizationNode].ToString() as [OrganizationNode], p.[FirstName] AS
    'ManagerFirstName', p.[LastName] AS 'ManagerLastName',
    [EMP_cte].[BusinessEntityID], [EMP_cte].[FirstName], [EMP_cte].[LastName] -- Outer select from the CTE
    FROM [EMP_cte]
        INNER JOIN [HumanResources].[Employee] e
            ON [EMP_cte].[OrganizationNode].GetAncestor(1) = e.[OrganizationNode]
                INNER JOIN [Person].[Person] p
                    ON p.[BusinessEntityID] = e.[BusinessEntityID]
    ORDER BY [RecursionLevel], [EMP_cte].[OrganizationNode].ToString()
    OPTION (MAXRECURSION 25)
END;

```

### 8.3.3.3. Procedure: dbo.uspSearchCandidateResumes

#### Input/Output

	Name	Data type	Description
@	searchString	nvarchar(1000)	
@	useInflectional	bit	
@	useThesaurus	bit	
@	language	int	

#### Uses

	Name
	dbo.uspSearchCandidateResumes
	HumanResources.JobCandidate

#### Script

```
--A stored procedure which demonstrates integrated full text search
CREATE PROCEDURE [dbo].[uspSearchCandidateResumes]
    @searchString nvarchar(1000),
    @useInflectional bit=0,
    @useThesaurus bit=0,
    @language int=0
WITH EXECUTE AS CALLER
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @string nvarchar(1050)
    --setting the lcid to the default instance LCID if needed
    IF @language = NULL OR @language = 0
    BEGIN
        SELECT @language = CONVERT(int, SERVERPROPERTY('lcid'))
    END
    --FREETEXTTABLE case as inflectional and Thesaurus were required
    IF @useThesaurus = 1 AND @useInflectional = 1
    BEGIN
        SELECT FT_TBL.[JobCandidateID], KEY_TBL.[RANK] FROM [HumanResources].[JobCandidate] AS FT_TBL
            INNER JOIN FREETEXTTABLE([HumanResources].[JobCandidate],*, @searchString, LANGUAGE @language) AS KEY_TBL
                ON FT_TBL.[JobCandidateID] = KEY_TBL.[KEY]
    END
    ELSE IF @useThesaurus = 1
    BEGIN
        SELECT @string = 'FORMSOF(THESAURUS,"'+@searchString +'''+'')
        SELECT FT_TBL.[JobCandidateID], KEY_TBL.[RANK] FROM [HumanResources].[JobCandidate] AS FT_TBL
            INNER JOIN CONTAINSTABLE([HumanResources].[JobCandidate],*, @string, LANGUAGE @language) AS KEY_TBL
                ON FT_TBL.[JobCandidateID] = KEY_TBL.[KEY]
    END
    ELSE IF @useInflectional = 1
    BEGIN
        SELECT @string = 'FORMSOF(INFLECTIONAL,"'+@searchString +'''+'')
        SELECT FT_TBL.[JobCandidateID], KEY_TBL.[RANK] FROM [HumanResources].[JobCandidate] AS FT_TBL
            INNER JOIN CONTAINSTABLE([HumanResources].[JobCandidate],*, @string, LANGUAGE @language) AS KEY_TBL
                ON FT_TBL.[JobCandidateID] = KEY_TBL.[KEY]
    END
    ELSE --base case, plain CONTAINSTABLE
    BEGIN
        SELECT @string=''''+@searchString +'''
        SELECT FT_TBL.[JobCandidateID],KEY_TBL.[RANK] FROM [HumanResources].[JobCandidate] AS FT_TBL
            INNER JOIN CONTAINSTABLE([HumanResources].[JobCandidate],*,@string, LANGUAGE @language) AS KEY_TBL
                ON FT_TBL.[JobCandidateID] = KEY_TBL.[KEY]
    END
END;
```

#### 8.3.3.4. Procedure: HumanResources.uspUpdateEmployeeHireInfo

##### Input/Output

	Name	Data type	Description
→@	BusinessEntityID	int	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a valid BusinessEntityID from the Employee table.
→@	JobTitle	nvarchar(50)	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a title for the employee.
→@	HireDate	datetime	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a hire date for the employee.
→@	RateChangeDate	datetime	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter the date the rate changed for the employee.
→@	Rate	money	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter the new rate for the employee.
→@	PayFrequency	tinyint	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter the pay frequency for the employee.
→@	CurrentFlag	bit	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter the current flag for the employee.

##### Uses

	Name
⚙️	HumanResources.uspUpdateEmployeeHireInfo
📄	HumanResources.Employee
📄	HumanResources.EmployeePayHistory
⚙️	dbo.uspLogError

##### Script

```

CREATE PROCEDURE [HumanResources].[uspUpdateEmployeeHireInfo]
    @BusinessEntityID [int],
    @JobTitle [nvarchar](50),
    @HireDate [datetime],
    @RateChangeDate [datetime],
    @Rate [money],
    @PayFrequency [tinyint],
    @CurrentFlag [dbo].[Flag]
WITH EXECUTE AS CALLER
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;
        UPDATE [HumanResources].[Employee]
        SET [JobTitle] = @JobTitle
            ,[HireDate] = @HireDate
            ,[CurrentFlag] = @CurrentFlag
        WHERE [BusinessEntityID] = @BusinessEntityID;
        INSERT INTO [HumanResources].[EmployeePayHistory]
        ([BusinessEntityID]
        ,[RateChangeDate]
        ,[Rate]
        ,[PayFrequency])
        VALUES (@BusinessEntityID, @RateChangeDate, @Rate, @PayFrequency);
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Rollback any active or uncommittable transactions before
        -- inserting information in the ErrorLog
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END
        EXECUTE [dbo].[uspLogError];
    END CATCH;
END;

```

### 8.3.3.5. Procedure: HumanResources.uspUpdateEmployeeLogin

Updates the Employee table with the values specified in the input parameters for the given BusinessEntityID.

#### Input/Output

	Name	Data type	Description
→@	BusinessEntityID	int	Input parameter for the stored procedure uspUpdateEmployeeLogin. Enter a valid EmployeeID from the Employee table.
→@	OrganizationNode	hierarchyid	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a valid ManagerID for the employee.
→@	LoginID	nvarchar(256)	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a valid login for the employee.
→@	JobTitle	nvarchar(50)	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a title for the employee.
→@	HireDate	datetime	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a hire date for the employee.
→@	CurrentFlag	bit	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter the current flag for the employee.

#### Uses

Name
HumanResources.uspUpdateEmployeeLogin
HumanResources.Employee
dbo.uspLogError

#### Script

```

CREATE PROCEDURE [HumanResources].[uspUpdateEmployeeLogin]
    @BusinessEntityID [int],
    @OrganizationNode [hierarchyid],
    @LoginID [nvarchar](256),
    @JobTitle [nvarchar](50),
    @HireDate [datetime],
    @CurrentFlag [dbo].[Flag]
WITH EXECUTE AS CALLER
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        UPDATE [HumanResources].[Employee]
        SET [OrganizationNode] = @OrganizationNode
        , [LoginID] = @LoginID
        , [JobTitle] = @JobTitle
        , [HireDate] = @HireDate
        , [CurrentFlag] = @CurrentFlag
        WHERE [BusinessEntityID] = @BusinessEntityID;
    END TRY
    BEGIN CATCH
        EXECUTE [dbo].[uspLogError];
    END CATCH;
END;

```

### 8.3.3.6. Procedure: HumanResources.uspUpdateEmployeePersonalInfo

Updates the Employee table with the values specified in the input parameters for the given EmployeeID.

#### Input/Output

	Name	Data type	Description
→@	BusinessEntityID	int	Input parameter for the stored procedure uspUpdateEmployeePersonalInfo. Enter a valid BusinessEntityID from the HumanResources.Employee table.
→@	NationalIDNumber	nvarchar(15)	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a national ID for the employee.
→@	BirthDate	datetime	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a birth date for the employee.
→@	MaritalStatus	nchar(1)	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a marital status for the employee.
→@	Gender	nchar(1)	Input parameter for the stored procedure uspUpdateEmployeeHireInfo. Enter a gender for the employee.

#### Uses

	Name
⚙️	HumanResources.uspUpdateEmployeePersonalInfo
💻	HumanResources.Employee
⚙️	dbo.uspLogError

#### Script

```
CREATE PROCEDURE [HumanResources].[uspUpdateEmployeePersonalInfo]
    @BusinessEntityID [int],
    @NationalIDNumber [nvarchar](15),
    @BirthDate [datetime],
    @MaritalStatus [nchar](1),
    @Gender [nchar](1)
WITH EXECUTE AS CALLER
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        UPDATE [HumanResources].[Employee]
        SET [NationalIDNumber] = @NationalIDNumber
            ,[BirthDate] = @BirthDate
            ,[MaritalStatus] = @MaritalStatus
            ,[Gender] = @Gender
        WHERE [BusinessEntityID] = @BusinessEntityID;
    END TRY
    BEGIN CATCH
        EXECUTE [dbo].[uspLogError];
    END CATCH;
END;
```

## 8.3.4. Functions

### 8.3.4.1. Function: dbo.ufnGetContactInformation

Table value function returning the first name, last name, job title and contact type for a given contact.

#### Input/Output

	Name	Data type	Description
*@	Returns	table type	
*@	PersonID	int	Input parameter for the table value function ufnGetContactInformation. Enter a valid PersonID from the Person.Contact table.

#### Uses

	Name
fx	dbo.ufnGetContactInformation
	HumanResources.Employee
	Person.BusinessEntityContact
	Person.ContactType
	Person.Person
	Purchasing.Vendor
	Sales.Customer
	Sales.Store

## Script

```
CREATE FUNCTION [dbo].[ufnGetContactInformation](@PersonID int)
RETURNS @retContactInformation TABLE
(
    -- Columns returned by the function
    [PersonID] int NOT NULL,
    [FirstName] [nvarchar](50) NULL,
    [LastName] [nvarchar](50) NULL,
    [JobTitle] [nvarchar](50) NULL,
    [BusinessEntityType] [nvarchar](50) NULL
)
AS
-- Returns the first name, last name, job title and business entity type for the specified contact.
-- Since a contact can serve multiple roles, more than one row may be returned.
BEGIN
    IF @PersonID IS NOT NULL
        BEGIN
            IF EXISTS(SELECT * FROM [HumanResources].[Employee] e
                      WHERE e.[BusinessEntityID] = @PersonID)
                INSERT INTO @retContactInformation
                    SELECT @PersonID, p.FirstName, p.LastName, e.[JobTitle], 'Employee'
                    FROM [HumanResources].[Employee] AS e
                        INNER JOIN [Person].[Person] p
                            ON p.[BusinessEntityID] = e.[BusinessEntityID]
                        WHERE e.[BusinessEntityID] = @PersonID;
            IF EXISTS(SELECT * FROM [Purchasing].[Vendor] AS v
                      INNER JOIN [Person].[BusinessEntityContact] bec
                          ON bec.[BusinessEntityID] = v.[BusinessEntityID]
                          WHERE bec.[PersonID] = @PersonID)
                INSERT INTO @retContactInformation
                    SELECT @PersonID, p.FirstName, p.LastName, ct.[Name], 'Vendor Contact'
                    FROM [Purchasing].[Vendor] AS v
                        INNER JOIN [Person].[BusinessEntityContact] bec
                            ON bec.[BusinessEntityID] = v.[BusinessEntityID]
                            INNER JOIN [Person].ContactType ct
                                ON ct.[ContactTypeID] = bec.[ContactTypeID]
                                INNER JOIN [Person].[Person] p
                                    ON p.[BusinessEntityID] = bec.[PersonID]
                                    WHERE bec.[PersonID] = @PersonID;
            IF EXISTS(SELECT * FROM [Sales].[Store] AS s
                      INNER JOIN [Person].[BusinessEntityContact] bec
                          ON bec.[BusinessEntityID] = s.[BusinessEntityID]
                          WHERE bec.[PersonID] = @PersonID)
                INSERT INTO @retContactInformation
                    SELECT @PersonID, p.FirstName, p.LastName, ct.[Name], 'Store Contact'
                    FROM [Sales].[Store] AS s
                        INNER JOIN [Person].[BusinessEntityContact] bec
                            ON bec.[BusinessEntityID] = s.[BusinessEntityID]
                            INNER JOIN [Person].ContactType ct
                                ON ct.[ContactTypeID] = bec.[ContactTypeID]
                                INNER JOIN [Person].[Person] p
                                    ON p.[BusinessEntityID] = bec.[PersonID]
                                    WHERE bec.[PersonID] = @PersonID;
            IF EXISTS(SELECT * FROM [Person].[Person] AS p
                      INNER JOIN [Sales].[Customer] AS c
                          ON c.[PersonID] = p.[BusinessEntityID]
                          WHERE p.[BusinessEntityID] = @PersonID AND c.[StoreID] IS NULL)
                INSERT INTO @retContactInformation
                    SELECT @PersonID, p.FirstName, p.LastName, NULL, 'Consumer'
                    FROM [Person].[Person] AS p
                        INNER JOIN [Sales].[Customer] AS c
                            ON c.[PersonID] = p.[BusinessEntityID]
                            WHERE p.[BusinessEntityID] = @PersonID AND c.[StoreID] IS NULL;
        END
    RETURN;
END;
```

TRIAL

## 8.4. Products



Products manufactured and sold by Adventure Works Cycles.

## 8.4.1. Tables

### 8.4.1.1. Table: Production.Culture

**Sample field: Status:** Active

Lookup table containing the languages in which some AdventureWorks data is stored.

#### Columns

	Name	Data type	Description / Attributes
☰	CultureID	nchar(6)	Primary key for Culture records.
☰	Name	Name: nvarchar(50)	Culture description.
☰	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

	Table	Join	Title / Name / Description
→	Production.ProductModelProductDescriptionCulture	<b>Production.Culture</b> CultureID = Production.ProductModelProductDescriptionCultureCultureID	FK_ProductModelProductDescriptionCulture_Culture_CultureID Foreign key constraint referencing Culture.CultureID.

#### Unique keys

	Columns	Name / Description
🔑	CultureID	PK_Culture_CultureID Primary key (clustered) constraint
🔑	Name	AK_Culture_Name Unique nonclustered index.

#### Used By

	Name
☰	Production.Culture
→	Production.ProductModelProductDescriptionCulture

## 8.4.1.2. Table: Production.Document

**Sample field: Status:** Active

Product maintenance documents.

### Columns

	Name	Data type	Description / Attributes
█	DocumentNode	hierarchyid	Primary key for Document records.
█	DocumentLevel	smallint	Depth in the document hierarchy. <b>Nullable</b> <b>Computed:</b> ([DocumentNode].[GetLevel]())
█	Title	nvarchar(50)	Title of the document.
█	Owner	int	Employee who controls the document. Foreign key to Employee.BusinessEntityID
█	FolderFlag	bit	0 = This is a folder, 1 = This is a document. <b>Default:</b> 0
█	FileName	nvarchar(400)	File name of the document
█	FileExtension	nvarchar(8)	File extension indicating the document type. For example, .doc or .txt.
█	Revision	nchar(5)	Revision number of the document.
█	ChangeNumber	int	Engineering change approval number. <b>Default:</b> 0
█	Status	tinyint	1 = Pending approval, 2 = Approved, 3 = Obsolete
█	DocumentSummary	nvarchar(MAX)	Document abstract. <b>Nullable</b>
█	Document	varbinary(MAX)	Complete document. <b>Nullable</b>
█	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Required for FileStream. <b>Default:</b> newid()
█	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

	Table	Join	Title / Name / Description
→	Production.ProductDocument	<b>Production.Document</b> DocumentNode = Production.ProductDocumentDocumentNode	FK_ProductDocument_Document_DocumentNode Foreign key constraint referencing Document.DocumentNode.

### Unique keys

	Columns	Name / Description
█	DocumentNode	PK_Document_DocumentNode Primary key (clustered) constraint
█	DocumentLevel, DocumentNode	AK_Document_DocumentLevel_DocumentNode Unique nonclustered index.
█	rowguid	AK_Document_rowguid Unique nonclustered index. Used to support FileStream.
█	rowguid	UQ_Document_F73921F793071A63

Uses

Name
Production.Document
Production.Document

Used By

Name
Production.Document
Production.Document
Production.ProductDocument

TRIAL

### 8.4.1.3. Table: Production.Illustration

**Sample field: Status:** Active

Bicycle assembly diagrams.

#### Columns

		Name	Data type	Description / Attributes
		IllustrationID	int	Primary key for Illustration records. <b>Identity / Auto increment</b>
		Diagram	xml	Illustrations used in manufacturing instructions. Stored as XML. <b>Nullable</b>
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

Table		Join	Title / Name / Description
	Production.ProductModelIllustration	<b>Production.Illustration</b> IllustrationID = Production.ProductModelIllustrationIllustrationID	FK_ProductModelIllustration_Illustration_IllustrationID Foreign key constraint referencing Illustration.IllustrationID.

#### Unique keys

Columns		Name / Description
	IllustrationID	PK_Illustration_IllustrationID Primary key (clustered) constraint

#### Used By

Name	
	Production.Illustration
	Production.ProductModelIllustration

#### 8.4.14. Table: Production.Product

**Sample field: Status:** Active

Products sold or used in the manufacturing of sold products.

##### Columns

	Name	Data type	Description / Attributes
■	ProductID	int	Primary key for Product records. <b>Identity / Auto increment</b>
■	Name	Name: nvarchar(50)	Name of the product.
■	ProductNumber	nvarchar(25)	Unique product identification number.
■	MakeFlag	Flag: bit	0 = Product is purchased, 1 = Product is manufactured in-house. <b>Default:</b> 1
■	FinishedGoodsFlag	Flag: bit	0 = Product is not a salable item. 1 = Product is salable. <b>Default:</b> 1
■	Color	nvarchar(15)	Product color. <b>Nullable</b>
■	SafetyStockLevel	smallint	Minimum inventory quantity.
■	ReorderPoint	smallint	Inventory level that triggers a purchase order or work order.
■	StandardCost	money	Standard cost of the product.
■	ListPrice	money	Selling price.
■	Size	nvarchar(5)	Product size. <b>Nullable</b>
■	SizeUnitMeasureCode	nchar(3)	Unit of measure for Size column. <b>Nullable</b> <b>References:</b> Production.UnitMeasure
■	WeightUnitMeasureCode	nchar(3)	Unit of measure for Weight column. <b>Nullable</b> <b>References:</b> Production.UnitMeasure
■	Weight	decimal(8, 2)	Product weight. <b>Nullable</b>
■	DaysToManufacture	int	Number of days required to manufacture the product.
■	ProductLine	nchar(2)	R = Road, M = Mountain, T = Touring, S = Standard <b>Nullable</b>
■	Class	nchar(2)	H = High, M = Medium, L = Low <b>Nullable</b>
■	Style	nchar(2)	W = Womens, M = Mens, U = Universal <b>Nullable</b>
■	ProductSubcategoryID	int	Product is a member of this product subcategory. Foreign key to ProductSubCategory.ProductSubCategoryID. <b>Nullable</b> <b>References:</b> Production.ProductSubcategory
■	ProductModelID	int	Product is a member of this product model. Foreign key to ProductModel.ProductModelID. <b>Nullable</b> <b>References:</b> Production.ProductModel
■	SellStartDate	datetime	Date the product was available for sale.
■	SellEndDate	datetime	Date the product was no longer available for sale. <b>Nullable</b>

Name		Data type	Description / Attributes
	DiscontinuedDate	datetime	Date the product was discontinued. <b>Nullable</b>
	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

## Links to

	Table	Join	Title / Name / Description
→	Production.ProductModel	<b>Production.Product</b> ProductModelID = Production.ProductModelProductModelID	FK_Product_ProductModel_ProductModelID Foreign key constraint referencing ProductModel.ProductModelID.
→	Production.ProductSubcategory	<b>Production.Product</b> ProductSubcategoryID = Production.ProductSubcategoryProductSubcategoryID	FK_Product_ProductSubcategory_ProductSubcategoryID Foreign key constraint referencing ProductSubcategory.ProductSubcategoryID.
→	Production.UnitMeasure	<b>Production.Product</b> SizeUnitMeasureCode = Production.UnitMeasureUnitMeasureCode	FK_Product_UnitMeasure_SizeUnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.
→	Production.UnitMeasure	<b>Production.Product</b> WeightUnitMeasureCode = Production.UnitMeasureUnitMeasureCode	FK_Product_UnitMeasure_WeightUnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.

## Linked from

	Table	Join	Title / Name / Description
←	Production.BillOfMaterials	<b>Production.Product</b> ProductID = Production.BillOfMaterialsComponentID	FK_BillOfMaterials_Product_ComponentID Foreign key constraint referencing Product.ProductAssemblyID.
←	Production.BillOfMaterials	<b>Production.Product</b> ProductID = Production.BillOfMaterialsProductAssemblyID	FK_BillOfMaterials_Product_ProductAssemblyID Foreign key constraint referencing Product.ProductAssemblyID.
←	Production.ProductCostHistory	<b>Production.Product</b> ProductID = Production.ProductCostHistoryProductID	FK_ProductCostHistory_Product_ProductID Foreign key constraint referencing Product.ProductID.
←	Production.ProductDocument	<b>Production.Product</b> ProductID = Production.ProductDocumentProductID	FK_ProductDocument_Product_ProductID Foreign key constraint referencing Product.ProductID.
←	Production.ProductInventory	<b>Production.Product</b> ProductID = Production.ProductInventoryProductID	FK_ProductInventory_Product_ProductID Foreign key constraint referencing Product.ProductID.
←	Production.ProductListPriceHistory	<b>Production.Product</b> ProductID = Production.ProductListPriceHistoryProductID	FK_ProductListPriceHistory_Product_ProductID Foreign key constraint referencing Product.ProductID.
←	Production.ProductProductPhoto	<b>Production.Product</b> ProductID = Production.ProductProductPhotoProductID	FK_ProductProductPhoto_Product_ProductID Foreign key constraint referencing Product.ProductID.
←	Production.ProductReview	<b>Production.Product</b> ProductID = Production.ProductReviewProductID	FK_ProductReview_Product_ProductID Foreign key constraint referencing Product.ProductID.
←	Purchasing.ProductVendor	<b>Production.Product</b> ProductID = Purchasing.ProductVendorProductID	FK_ProductVendor_Product_ProductID Foreign key constraint referencing Product.ProductID.

	Table	Join	Title / Name / Description
→	Purchasing.PurchaseOrderDetail	<b>Production.Product</b> ProductID = Purchasing.PurchaseOrderDetailProductID	FK_PurchaseOrderDetail_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Sales.SalesOrderDetail	<b>Production.Product</b> ProductID = Sales.SalesOrderDetailProductID	User-defined relation
→	Sales.ShoppingCartItem	<b>Production.Product</b> ProductID = Sales.ShoppingCartItemProductID	FK_ShoppingCartItem_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Sales.SpecialOfferProduct	<b>Production.Product</b> ProductID = Sales.SpecialOfferProductProductID	FK_SpecialOfferProduct_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Production.TransactionHistory	<b>Production.Product</b> ProductID = Production.TransactionHistoryProductID	FK_TransactionHistory_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Production.TransactionHistoryArchive	<b>Production.Product</b> ProductID = Production.TransactionHistoryArchiveProductID	User-defined relation
→	Production.WorkOrder	<b>Production.Product</b> ProductID = Production.WorkOrderProductID	FK_WorkOrder_Product_ProductID Foreign key constraint referencing Product.ProductID.

## Unique keys

	Columns	Name / Description
🔑	ProductID	PK_Product_ProductID Primary key (clustered) constraint
🔑	Name	AK_Product_Name Unique nonclustered index.
🔑	ProductNumber	AK_Product_ProductNumber Unique nonclustered index.
🔑	rowguid	AK_Product_rowguid Unique nonclustered index. Used to support replication samples.

## Uses

	Name
grid	<b>Production.Product</b>
→	Production.ProductModel
→	Production.ProductSubcategory
→	Production.UnitMeasure
→	Production.UnitMeasure

## Used By

	Name
grid	<b>Production.Product</b>
file	Production.vProductAndDescription
gear	dbo.uspGetBillOfMaterials
gear	dbo.uspGetWhereUsedProductID
fx	dbo.ufnGetProductDealerPrice
fx	dbo.ufnGetProductListPrice
fx	dbo.ufnGetProductStandardCost

Name
→ Production.BillOfMaterials
→ Production.BillOfMaterials
→ Production.ProductCostHistory
→ Production.ProductDocument
→ Production.ProductInventory
→ Production.ProductListPriceHistory
→ Production.ProductProductPhoto
→ Production.ProductReview
→ Production.TransactionHistory
→ Production.TransactionHistoryArchive
→ Production.WorkOrder
→ Purchasing.ProductVendor
→ Purchasing.PurchaseOrderDetail
→ Sales.SalesOrderDetail
→ Sales.ShoppingCartItem
→ Sales.SpecialOfferProduct

TRIAL

## 8.4.1.5. Table: Production.ProductCategory

**Sample field: Status:** Active

High-level product categorization.

### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductCategoryID	int	Primary key for ProductCategory records. <b>Identity / Auto increment</b>
■	🔑	Name	Name: nvarchar(50)	Category description.
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
→	Production.ProductSubcategory	Production.ProductCategory ProductCategoryID = Production.ProductSubcategoryProductCategoryID	FK_ProductSubcategory_ProductCategory_ProductCategoryID Foreign key constraint referencing ProductCategory.ProductCategoryID.

### Unique keys

Columns		Name / Description
🔑	ProductCategoryID	PK_ProductCategory_ProductCategoryID Primary key (clustered) constraint
🔑	Name	AK_ProductCategory_Name Unique nonclustered index.
🔑	rowguid	AK_ProductCategory_rowguid Unique nonclustered index. Used to support replication samples.

### Used By

Name	
■	Production.ProductCategory
→	Production.ProductSubcategory

## 8.4.1.6. Table: Production.ProductDescription

**Sample field: Status:** Active

Product descriptions in several languages.

### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductDescriptionID	int	Primary key for ProductDescription records. <b>Identity / Auto increment</b>
■		Description	nvarchar(400)	Description of the product.
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
→	Production.ProductModelProductDescriptionCulture	Production.ProductDescriptionProductDescriptionID = Production.ProductModelProductDescriptionCultureProductDescriptionID	FK_ProductModelProductDescriptionCulture_ProductDescription_ProductDescriptionID Foreign key constraint referencing ProductDescription.ProductDescriptionID.

### Unique keys

Columns		Name / Description
🔑	ProductDescriptionID	PK_ProductDescription_ProductDescriptionID Primary key (clustered) constraint
🔑	rowguid	AK_ProductDescription_rowguid Unique nonclustered index. Used to support replication samples.

### Used By

Name	
■	Production.ProductDescription
↳	Production.vProductAndDescription
→	Production.ProductModelProductDescriptionCulture

#### 8.4.1.7. Table: Production.ProductDocument

**Sample field: Status:** Active

Cross-reference table mapping products to related product documents.

##### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
■	🔑	DocumentNode	hierarchyid	Document identification number. Foreign key to Document.DocumentNode. <b>References:</b> Production.Document
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

##### Links to

	Table	Join	Title / Name / Description
→	Production.Document	<b>Production.ProductDocument</b> DocumentNode = Production.DocumentDocumentNode	FK_ProductDocument_Document_DocumentNode Foreign key constraint referencing Document.DocumentNode.
→	Production.Product	<b>Production.ProductDocument</b> ProductID = Production.ProductProductID	FK_ProductDocument_Product_ProductID Foreign key constraint referencing Product.ProductID.

##### Unique keys

	Columns	Name / Description
🔑	ProductID, DocumentNode	PK_ProductDocument_ProductID_DocumentNode Primary key (clustered) constraint

##### Uses

	Name
grid	Production.ProductDocument
→	Production.Document
→	Production.Product

## 8.4.1.8. Table: Production.ProductModel

**Sample field: Status:** Active

Product model classification.

### Columns

		Name	Data type	Description / Attributes
	ProductModelID	int		Primary key for ProductModel records. <b>Identity / Auto increment</b>
	Name	Name: nvarchar(50)		Product model description.
	CatalogDescription	xml		Detailed product catalog information in xml format. <b>Nullable</b>
	Instructions	xml		Manufacturing instructions in xml format. <b>Nullable</b>
	rowguid	uniqueidentifier		ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
	ModifiedDate	datetime		Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

		Table	Join	Title / Name / Description
	Production.Product	<b>Production.ProductModel</b> ProductMod elID = Production.ProductProductModelID	FK_Product_ProductModel_ProductModelID	Foreign key constraint referencing ProductModel.ProductModelID.
	Production.ProductModelIllustratio n	<b>Production.ProductModel</b> ProductMod elID = Production.ProductModelIllustrationPr oductModelID	FK_ProductModelIllustration_ProductModel_ProductModelID	Foreign key constraint referencing ProductModel.ProductModelID.
	Production.ProductModelProductD escriptionCulture	<b>Production.ProductModel</b> ProductMod elID = Production.ProductModelProductDesc riptionCultureProductModelID	FK_ProductModelProductDescriptionCulture_ProductModel_ProductM odelID	Foreign key constraint referencing ProductModel.ProductModelID.

### Unique keys

		Columns	Name / Description
	ProductModelID	PK_ProductModel_ProductModelID	Primary key (clustered) constraint
	Name	AK_ProductModel_Name	Unique nonclustered index.
	rowguid	AK_ProductModel_rowguid	Unique nonclustered index. Used to support replication samples.

### Used By

		Name
	<b>Production.ProductModel</b>	
	Production.vProductAndDescription	
	Production.vProductModelCatalogDescription	
	Production.vProductModelInstructions	

Name

→ Production.Product

→ Production.ProductModelIllustration

→ Production.ProductModelProductDescriptionCulture

TRIAL

## 8.4.1.9. Table: Production.ProductModelIllustration

**Sample field: Status:** Active

Cross-reference table mapping product models and illustrations.

### Columns

Name		Data type	Description / Attributes
ProductModelID	Primary key. Foreign key to ProductModel.ProductModelID. <b>References:</b> Production.ProductModel	int	
IllustrationID	Primary key. Foreign key to Illustration.IllustrationID. <b>References:</b> Production.Illustration	int	
ModifiedDate	Date and time the record was last updated. <b>Default:</b> getdate()	datetime	

### Links to

Table		Join	Title / Name / Description
Production.Illustration		<b>Production.ProductModelIllustration</b> IllustrationID = Production.IllustrationIllustrationID	FK_ProductModelIllustration_Illustration_IllustrationID Foreign key constraint referencing Illustration.IllustrationID.
Production.ProductModel		<b>Production.ProductModelIllustration</b> ProductModelID = Production.ProductModelProductModelID	FK_ProductModelIllustration_ProductModel_ProductModelID Foreign key constraint referencing ProductModel.ProductModelID.

### Unique keys

Columns		Name / Description
ProductModelID, IllustrationID		PK_ProductModelIllustration_ProductModelID_IllustrationID Primary key (clustered) constraint

### Uses

Name	
Production.ProductModelIllustration	
Production.Illustration	
Production.ProductModel	

## 8.4.1.10. Table: Production.ProductModelProductDescriptionCulture

**Sample field: Status:** Active

Cross-reference table mapping product descriptions and the language the description is written in.

### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductModelID	int	Primary key. Foreign key to ProductModel.ProductModelID. <b>References:</b> Production.ProductModel
■	🔑	ProductDescriptionID	int	Primary key. Foreign key to ProductDescription.ProductDescriptionID. <b>References:</b> Production.ProductDescription
■	🔑	CultureID	nchar(6)	Culture identification number. Foreign key to Culture.CultureID. <b>References:</b> Production.Culture
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Culture	<b>Production.ProductModelProductDescriptionCulture</b> CultureID = Production.CultureCultureID	FK_ProductModelProductDescriptionCulture_Culture_CultureID Foreign key constraint referencing Culture.CultureID.
→	Production.ProductDescription	<b>Production.ProductModelProductDescriptionCulture</b> ProductDescriptionID = Production.ProductDescriptionProductDescriptionID	FK_ProductModelProductDescriptionCulture_ProductDescription_ProductDescriptionID Foreign key constraint referencing ProductDescription.ProductDescriptionID.
→	Production.ProductModel	<b>Production.ProductModelProductDescriptionCulture</b> ProductModelID = Production.ProductModelProductModelID	FK_ProductModelProductDescriptionCulture_ProductModel_ProductModelID Foreign key constraint referencing ProductModel.ProductModelID.

### Unique keys

Columns		Name / Description
🔑	ProductModelID, ProductDescriptionID, CultureID	PK_ProductModelProductDescriptionCulture_ProductModelID_ProductDescriptionID_CultureID Primary key (clustered) constraint

### Uses

		Name
■	Production.ProductModelProductDescriptionCulture	
→	Production.Culture	
→	Production.ProductDescription	
→	Production.ProductModel	

### Used By

		Name
■	Production.ProductModelProductDescriptionCulture	
↑	Production.vProductAndDescription	

## 8.4.1.11. Table: Production.ProductPhoto

**Sample field: Status:** Active

Product images.

### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductPhotoID	int	Primary key for ProductPhoto records. <b>Identity / Auto increment</b>
■		ThumbNailPhoto	varbinary(MAX)	Small image of the product. <b>Nullable</b>
■		ThumbnailPhotoFileName	nvarchar(50)	Small image file name. <b>Nullable</b>
■		LargePhoto	varbinary(MAX)	Large image of the product. <b>Nullable</b>
■		LargePhotoFileName	nvarchar(50)	Large image file name. <b>Nullable</b>
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

		Table	Join	Title / Name / Description
→	Production.ProductProductPhoto	Production.ProductPhoto	ProductPhotoID = Production.ProductProductPhotoProductPhotoID	FK_ProductProductPhoto_ProductPhoto_ProductPhotoID Foreign key constraint referencing ProductPhoto.ProductPhotoID.

### Unique keys

		Columns	Name / Description
🔑	ProductPhotoID	PK_ProductPhoto_ProductPhotoID	Primary key (clustered) constraint

### Used By

		Name
■	Production.ProductPhoto	
→	Production.ProductProductPhoto	

## 8.4.1.12. Table: Production.ProductProductPhoto

**Sample field: Status:** Active

Cross-reference table mapping products and product photos.

### Columns

Name			Data type	Description / Attributes
ProductID	ProductID	int		Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
ProductPhotoID	ProductPhotoID	int		Product photo identification number. Foreign key to ProductPhoto.ProductPhotoID. <b>References:</b> Production.ProductPhoto
Primary	Primary	Flag: bit		0 = Photo is not the principal image. 1 = Photo is the principal image. <b>Default:</b> 0
ModifiedDate	ModifiedDate	datetime		Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
Production.Product	Production.ProductProductPhoto	ProductID = Production.ProductProductID	FK_ProductProductPhoto_Product_ProductID Foreign key constraint referencing Product.ProductID.
Production.ProductPhoto	Production.ProductProductPhoto	ProductPhotoID = Production.ProductPhotoProductPhotoID	FK_ProductProductPhoto_ProductPhoto_ProductPhotoID Foreign key constraint referencing ProductPhoto.ProductPhotoID.

### Unique keys

Columns		Name / Description
ProductID, ProductPhotoID		PK_ProductProductPhoto_ProductID_ProductPhotoID Primary key (clustered) constraint

### Uses

Name	
Production.ProductProductPhoto	
Production.Product	
Production.ProductPhoto	

## 8.4.1.13. Table: Production.ProductReview

**Sample field: Status:** Active

Customer reviews of products they have purchased.

### Columns

		Name	Data type	Description / Attributes
grid	key	ProductReviewID	int	Primary key for ProductReview records. <b>Identity / Auto increment</b>
grid		ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
grid		ReviewerName	Name: nvarchar(50)	Name of the reviewer.
grid		ReviewDate	datetime	Date review was submitted. <b>Default:</b> getdate()
grid		EmailAddress	nvarchar(50)	Reviewer's e-mail address.
grid		Rating	int	Product rating given by the reviewer. Scale is 1 to 5 with 5 as the highest rating.
grid		Comments	nvarchar(3850)	Reviewer's comments <b>Nullable</b>
grid		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
grid	Production.Product	Production.ProductReview	ProductID = Production.ProductProductID Foreign key constraint referencing Product.ProductID.

### Unique keys

Columns		Name / Description
key	ProductReviewID	PK_ProductReview_ProductReviewID Primary key (clustered) constraint

### Uses

Name	
grid	Production.ProductReview
grid	Production.Product

#### 8.4.1.14. Table: Production.ProductSubcategory

**Sample field: Status:** Active

Product subcategories. See ProductCategory table.

##### Columns

		Name	Data type	Description / Attributes
		ProductSubcategoryID	int	Primary key for ProductSubcategory records. <b>Identity / Auto increment</b>
		ProductCategoryID	int	Product category identification number. Foreign key to ProductCategory.ProductCategoryID. <b>References:</b> Production.ProductCategory
		Name	Name: nvarchar(50)	Subcategory description.
		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

##### Links to

Table		Join	Title / Name / Description
	Production.ProductCategory	Production.ProductSubcategoryProductCategoryID = Production.ProductCategoryProductCategoryID	FK_ProductSubcategory_ProductCategory_ProductCategoryID Foreign key constraint referencing ProductCategory.ProductCategoryID.

##### Linked from

Table		Join	Title / Name / Description
	Production.Product	Production.ProductSubcategoryProductSubcategoryID = Production.ProductProductSubcategoryID	FK_Product_ProductSubcategory_ProductSubcategoryID Foreign key constraint referencing ProductSubcategory.ProductSubcategoryID.

##### Unique keys

Columns		Name / Description
	ProductSubcategoryID	PK_ProductSubcategory_ProductSubcategoryID Primary key (clustered) constraint
	Name	AK_ProductSubcategory_Name Unique nonclustered index.
	rowguid	AK_ProductSubcategory_rowguid Unique nonclustered index. Used to support replication samples.

##### Uses

		Name
	Production.ProductSubcategory	
	Production.ProductCategory	

##### Used By

		Name
	Production.ProductSubcategory	

TRIAL

## 8.4.1.15. Table: Production.UnitMeasure

**Sample field: Status:** Active

Unit of measure lookup table.

### Columns

		Name	Data type	Description / Attributes
		UnitMeasureCode	nchar(3)	Primary key.
		Name	Name: nvarchar(50)	Unit of measure description.
		ModifiedDate	datetime	Date and time the record was last updated. Default: getdate()

### Linked from

	Table	Join	Title / Name / Description
→	Production.BillOfMaterials	<b>Production.UnitMeasure</b> UnitMeasureCode = Production.BillOfMaterialsUnitMeasureCode	FK_BillOfMaterials_UnitMeasure_UnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.
→	Production.Product	<b>Production.UnitMeasure</b> UnitMeasureCode = Production.ProductSizeUnitMeasureCode	FK_Product_UnitMeasure_SizeUnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.
→	Production.Product	<b>Production.UnitMeasure</b> UnitMeasureCode = Production.ProductWeightUnitMeasureCode	FK_Product_UnitMeasure_WeightUnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.
→	Purchasing.ProductVendor	<b>Production.UnitMeasure</b> UnitMeasureCode = Purchasing.ProductVendorUnitMeasureCode	FK_ProductVendor_UnitMeasure_UnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.

### Unique keys

	Columns	Name / Description
	UnitMeasureCode	PK_UnitMeasure_UnitMeasureCode Primary key (clustered) constraint
	Name	AK_UnitMeasure_Name Unique nonclustered index.

### Used By

	Name
	<b>Production.UnitMeasure</b>
→	Production.BillOfMaterials
→	Production.Product
→	Production.Product
→	Purchasing.ProductVendor

## 8.4.2. Views

### 8.4.2.1. View: Production.vProductAndDescription

**Sample field: Status:** Active

Product names and descriptions. Product descriptions are provided in multiple languages.

#### Columns

	Name	Data type	Description / Attributes
■	ProductID	int	
■	Name	Name: nvarchar(50)	
■	ProductModel	Name: nvarchar(50)	
■	CultureID	nchar(6)	
■	Description	nvarchar(400)	

#### Unique keys

	Columns	Name / Description
■	CultureID, ProductID	IX_vProductAndDescription Clustered index on the view vProductAndDescription.

#### Uses

	Name
■	Production.vProductAndDescription
■	Production.Product
■	Production.ProductDescription
■	Production.ProductModel
■	Production.ProductModelProductDescriptionCulture

#### Script

```
CREATE VIEW [Production].[vProductAndDescription]
WITH SCHEMABINDING
AS
-- View (indexed or standard) to display products and product descriptions by language.
SELECT
    p.[ProductID]
    ,p.[Name]
    ,pm.[Name] AS [ProductModel]
    ,pmx.[CultureID]
    ,pd.[Description]
FROM [Production].[Product] p
INNER JOIN [Production].[ProductModel] pm
ON p.[ProductModelID] = pm.[ProductModelID]
INNER JOIN [Production].[ProductModelProductDescriptionCulture] pmx
ON pm.[ProductModelID] = pmx.[ProductModelID]
INNER JOIN [Production].[ProductDescription] pd
ON pmx.[ProductDescriptionID] = pd.[ProductDescriptionID];
```

## 8.4.2.2. View: Production.vProductModelCatalogDescription

### Sample field: Status: Active

Displays the content from each element in the xml column CatalogDescription for each product in the Production.ProductModel table that has catalog data.

### Columns

	Name	Data type	Description / Attributes
1	ProductModelID	int	Identity / Auto increment
2	Name	Name: nvarchar(50)	
3	Summary	nvarchar(MAX)	Nullable
4	Manufacturer	nvarchar(MAX)	Nullable
5	Copyright	nvarchar(30)	Nullable
6	ProductURL	nvarchar(256)	Nullable
7	WarrantyPeriod	nvarchar(256)	Nullable
8	WarrantyDescription	nvarchar(256)	Nullable
9	NoOfYears	nvarchar(256)	Nullable
10	MaintenanceDescription	nvarchar(256)	Nullable
11	Wheel	nvarchar(256)	Nullable
12	Saddle	nvarchar(256)	Nullable
13	Pedal	nvarchar(256)	Nullable
14	BikeFrame	nvarchar(MAX)	Nullable
15	Crankset	nvarchar(256)	Nullable
16	PictureAngle	nvarchar(256)	Nullable
17	PictureSize	nvarchar(256)	Nullable
18	ProductPhotoID	nvarchar(256)	Nullable
19	Material	nvarchar(256)	Nullable
20	Color	nvarchar(256)	Nullable
21	ProductLine	nvarchar(256)	Nullable
22	Style	nvarchar(256)	Nullable
23	RiderExperience	nvarchar(1024)	Nullable
24	rowguid	uniqueidentifier	
25	ModifiedDate	datetime	

### Uses

Name
Production.vProductModelCatalogDescription
Production.ProductModel

## Script

```
CREATE VIEW [Production].[vProductModelCatalogDescription]
AS
SELECT
    [ProductModelID]
    ,[Name]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace html="http://www.w3.org/1999/xhtml";
        (/p1:ProductDescription/p1:Summary/html:p)[1]', 'nvarchar(max)' ) AS [Summary]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Manufacturer/p1:Name)[1]', 'nvarchar(max)' ) AS [Manufacturer]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Manufacturer/p1:Copyright)[1]', 'nvarchar(30)' ) AS [Copyright]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Manufacturer/p1:ProductURL)[1]', 'nvarchar(256)' ) AS [ProductURL]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wm="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelWarrAndMain";
        (/p1:ProductDescription/p1:Features/wm:Warranty/wm:WarrantyPeriod)[1]', 'nvarchar(256)' ) AS [WarrantyPeriod]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wm="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelWarrAndMain";
        (/p1:ProductDescription/p1:Features/wm:Warranty/wm:Description)[1]', 'nvarchar(256)' ) AS [WarrantyDescription]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wm="http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelWarrAndMain";
        (/p1:ProductDescription/p1:Features/wm:Maintenance/wm:NoOfYears)[1]', 'nvarchar(256)' ) AS [NoOfYears]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wf="http://www.adventure-works.com/schemas/OtherFeatures";
        (/p1:ProductDescription/p1:Features/wf:wheel)[1]', 'nvarchar(256)' ) AS [Wheel]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wf="http://www.adventure-works.com/schemas/OtherFeatures";
        (/p1:ProductDescription/p1:Features/wf:saddle)[1]', 'nvarchar(256)' ) AS [Saddle]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wf="http://www.adventure-works.com/schemas/OtherFeatures";
        (/p1:ProductDescription/p1:Features/wf:pedal)[1]', 'nvarchar(256)' ) AS [Pedal]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wf="http://www.adventure-works.com/schemas/OtherFeatures";
        (/p1:ProductDescription/p1:Features/wf:BikeFrame)[1]', 'nvarchar(max)' ) AS [BikeFrame]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        declare namespace wf="http://www.adventure-works.com/schemas/OtherFeatures";
        (/p1:ProductDescription/p1:Features/wf:crankset)[1]', 'nvarchar(256)' ) AS [Crankset]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Picture/p1:Angle)[1]', 'nvarchar(256)' ) AS [PictureAngle]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Picture/p1:Size)[1]', 'nvarchar(256)' ) AS [PictureSize]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Picture/p1:ProductPhotoID)[1]', 'nvarchar(256)' ) AS [ProductPhotoID]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Specifications/Material)[1]', 'nvarchar(256)' ) AS [Material]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Specifications/Color)[1]', 'nvarchar(256)' ) AS [Color]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Specifications/ProductLine)[1]', 'nvarchar(256)' ) AS [ProductLine]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Specifications/Style)[1]', 'nvarchar(256)' ) AS [Style]
    ,[CatalogDescription].value(N'declare namespace p1="http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelDescription";
        (/p1:ProductDescription/p1:Specifications/RiderExperience)[1]', 'nvarchar(1024)' ) AS [RiderExperience]
    ,[rowguid]
    ,[ModifiedDate]
FROM [Production].[ProductModel]
WHERE [CatalogDescription] IS NOT NULL;
```

### 8.4.2.3. View: Production.vProductModelInstructions

#### Sample field: Status: Active

Displays the content from each element in the xml column Instructions for each product in the Production.ProductModel table that has manufacturing instructions.

#### Columns

	Name	Data type	Description / Attributes
1	ProductModelID	int	Identity / Auto increment
2	Name	Name: nvarchar(50)	
3	Instructions	nvarchar(MAX)	Nullable
4	LocationID	int	Nullable
5	SetupHours	decimal(9, 4)	Nullable
6	MachineHours	decimal(9, 4)	Nullable
7	LaborHours	decimal(9, 4)	Nullable
8	LotSize	int	Nullable
9	Step	nvarchar(1024)	Nullable
10	rowguid	uniqueidentifier	
11	ModifiedDate	datetime	

#### Uses

	Name
1	Production.vProductModelInstructions
2	Production.ProductModel
3	MfgInstructions.ref.value
4	Steps.ref.value

#### Script

```

CREATE VIEW [Production].[vProductModelInstructions]
AS
SELECT
    [ProductModelID]
    ,[Name]
    , [Instructions].value(N'declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/ProductModelManuInstructions";
        (/root/text())[1]', 'nvarchar(max)' ) AS [Instructions]
    , [MfgInstructions].ref.value('@LocationID[1]', 'int') AS [LocationID]
    , [MfgInstructions].ref.value('@SetupHours[1]', 'decimal(9, 4)') AS [SetupHours]
    , [MfgInstructions].ref.value('@MachineHours[1]', 'decimal(9, 4)') AS [MachineHours]
    , [MfgInstructions].ref.value('@LaborHours[1]', 'decimal(9, 4)') AS [LaborHours]
    , [MfgInstructions].ref.value('@LotSize[1]', 'int') AS [LotSize]
    , [Steps].ref.value('string())[1]', 'nvarchar(1024)' ) AS [Step]
    , [rowguid]
    , [ModifiedDate]
FROM [Production].[ProductModel]
CROSS APPLY [Instructions].nodes(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelManuInstructions";
        /root/Location') MfgInstructions(ref)
CROSS APPLY [MfgInstructions].ref.nodes('declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelManuInstructions";
        step') Steps(ref);

```

## 8.4.3. Procedures

### 8.4.3.1. Procedure: dbo.uspGetBillOfMaterials

Uses a recursive query (common table expression) to generate a multilevel Bill of Material: all level 1 components of a level 0 assembly, all level 2 components of a level 1 assembly, and so on.

#### Input/Output

	Name	Data type	Description
→@	StartProductID	int	Input parameter for the stored procedure uspGetBillOfMaterials. Enter a valid ProductID from the Production.Product table.
→@	CheckDate	datetime	Input parameter for the stored procedure uspGetBillOfMaterials used to eliminate components not used after that date. Enter a valid date.

#### Uses

Name
dbo.uspGetBillOfMaterials
Production.BillOfMaterials
Production.Product

#### Script

```
CREATE PROCEDURE [dbo].[uspGetBillOfMaterials]
    @StartProductID [int],
    @CheckDate [datetime]
AS
BEGIN
    SET NOCOUNT ON;
    -- Use recursive query to generate a multi-level Bill of Material (i.e. all level 1
    -- components of a level 0 assembly, all level 2 components of a level 1 assembly)
    -- The CheckDate eliminates any components that are no longer used in the product on this date.
    WITH [BOM_cte]([ProductAssemblyID], [ComponentID], [ComponentDesc], [PerAssemblyQty], [StandardCost], [ListPrice],
    [BOMLevel], [RecursionLevel]) -- CTE name and columns
    AS (
        SELECT b.[ProductAssemblyID], b.[ComponentID], p.[Name], b.[PerAssemblyQty], p.[StandardCost], p.[ListPrice],
        b.[BOMLevel], 0 -- Get the initial list of components for the bike assembly
        FROM [Production].[BillOfMaterials] b
        INNER JOIN [Production].[Product] p
        ON b.[ComponentID] = p.[ProductID]
        WHERE b.[ProductAssemblyID] = @StartProductID
        AND @CheckDate >= b.[StartDate]
        AND @CheckDate <= ISNULL(b.[EndDate], @CheckDate)
    UNION ALL
        SELECT b.[ProductAssemblyID], b.[ComponentID], p.[Name], b.[PerAssemblyQty], p.[StandardCost], p.[ListPrice],
        b.[BOMLevel], [RecursionLevel] + 1 -- Join recursive member to anchor
        FROM [BOM_cte] cte
        INNER JOIN [Production].[BillOfMaterials] b
        ON b.[ProductAssemblyID] = cte.[ComponentID]
        INNER JOIN [Production].[Product] p
        ON b.[ComponentID] = p.[ProductID]
        WHERE @CheckDate >= b.[StartDate]
        AND @CheckDate <= ISNULL(b.[EndDate], @CheckDate)
    )
    -- Outer select from the CTE
    SELECT b.[ProductAssemblyID], b.[ComponentID], b.[ComponentDesc], SUM(b.[PerAssemblyQty]) AS [TotalQuantity] ,
    b.[StandardCost], b.[ListPrice], b.[BOMLevel], b.[RecursionLevel]
    FROM [BOM_cte] b
    GROUP BY b.[ComponentID], b.[ComponentDesc], b.[ProductAssemblyID], b.[BOMLevel], b.[RecursionLevel], b.[StandardCost],
    b.[ListPrice]
    ORDER BY b.[BOMLevel], b.[ProductAssemblyID], b.[ComponentID]
    OPTION (MAXRECURSION 25)
END;
```

### 8.4.3.2. Procedure: dbo.uspGetWhereUsedProductID

Stored procedure using a recursive query to return all components or assemblies that directly or indirectly use the specified ProductID.

#### Input/Output

	Name	Data type	Description
→@	StartProductID	int	Input parameter for the stored procedure uspGetWhereUsedProductID. Enter a valid ProductID from the Production.Product table.
→@	CheckDate	datetime	Input parameter for the stored procedure uspGetWhereUsedProductID used to eliminate components not used after that date. Enter a valid date.

#### Uses

Name
dbo.uspGetWhereUsedProductID
Production.BillOfMaterials
Production.Product

#### Script

```

CREATE PROCEDURE [dbo].[uspGetWhereUsedProductID]
    @StartProductID [int],
    @CheckDate [datetime]
AS
BEGIN
    SET NOCOUNT ON;
    --Use recursive query to generate a multi-level Bill of Material (i.e. all level 1 components of a level 0 assembly, all
    level 2 components of a level 1 assembly)
    WITH [BOM_cte]([ProductAssemblyID], [ComponentID], [ComponentDesc], [PerAssemblyQty], [StandardCost], [ListPrice],
    [BOMLevel], [RecursionLevel]) -- CTE name and columns
    AS (
        SELECT b.[ProductAssemblyID], b.[ComponentID], p.[Name], b.[PerAssemblyQty], p.[StandardCost], p.[ListPrice],
        b.[BOMLevel], 0 -- Get the initial list of components for the bike assembly
        FROM [Production].[BillOfMaterials] b
        INNER JOIN [Production].[Product] p
        ON b.[ProductAssemblyID] = p.[ProductID]
        WHERE b.[ComponentID] = @StartProductID
        AND @CheckDate >= b.[StartDate]
        AND @CheckDate <= ISNULL(b.[EndDate], @CheckDate)
    UNION ALL
        SELECT b.[ProductAssemblyID], b.[ComponentID], p.[Name], b.[PerAssemblyQty], p.[StandardCost], p.[ListPrice],
        b.[BOMLevel], [RecursionLevel] + 1 -- Join recursive member to anchor
        FROM [BOM_cte] cte
        INNER JOIN [Production].[BillOfMaterials] b
        ON cte.[ProductAssemblyID] = b.[ComponentID]
        INNER JOIN [Production].[Product] p
        ON b.[ProductAssemblyID] = p.[ProductID]
        WHERE @CheckDate >= b.[StartDate]
        AND @CheckDate <= ISNULL(b.[EndDate], @CheckDate)
    )
    -- Outer select from the CTE
    SELECT b.[ProductAssemblyID], b.[ComponentID], b.[ComponentDesc], SUM(b.[PerAssemblyQty]) AS [TotalQuantity] ,
    [StandardCost], b.[ListPrice], b.[BOMLevel], b.[RecursionLevel]
    FROM [BOM_cte] b
    GROUP BY b.[ComponentID], b.[ComponentDesc], b.[ProductAssemblyID], b.[BOMLevel], b.[RecursionLevel], b.[StandardCost],
    b.[ListPrice]
    ORDER BY b.[BOMLevel], b.[ProductAssemblyID], b.[ComponentID]
    OPTION (MAXRECURSION 25)
END;

```

## 8.4.4. Functions

### 8.4.4.1. Function: dbo.ufnGetDocumentStatusText

Scalar function returning the text representation of the Status column in the Document table.

#### Input/Output

	Name	Data type	Description
*@	Returns	nvarchar(16)	Returns the text representation of the Status column
*@	Status	tinyint	Input parameter for the scalar function ufnGetDocumentStatusText. Enter a valid integer.

#### Script

```
CREATE FUNCTION [dbo].[ufnGetDocumentStatusText] (@Status [tinyint])
RETURNS [nvarchar] (16)
AS
-- Returns the sales order status text representation for the status value.
BEGIN
    DECLARE @ret [nvarchar] (16);
    SET @ret =
        CASE @Status
            WHEN 1 THEN N'Pending approval'
            WHEN 2 THEN N'Approved'
            WHEN 3 THEN N'Obsolete'
            ELSE N'** Invalid **'
        END;
    RETURN @ret
END;
```

#### 8.4.4.2. Function: dbo.ufnGetProductDealerPrice

Scalar function returning the dealer price for a given product on a particular order date.

##### Input/Output

	Name	Data type	Description
*@	Returns	money	
*@	ProductID	int	Input parameter for the scalar function ufnGetProductDealerPrice. Enter a valid ProductID from the Production.Product table.
*@	OrderDate	datetime	Input parameter for the scalar function ufnGetProductDealerPrice. Enter a valid order date.

##### Uses

Name
dbo.ufnGetProductDealerPrice
Production.Product
Production.ProductListPriceHistory

##### Script

```
CREATE FUNCTION [dbo].[ufnGetProductDealerPrice] (@ProductID [int], @OrderDate [datetime])
RETURNS [money]
AS
-- Returns the dealer price for the product on a specific date.
BEGIN
    DECLARE @DealerPrice money;
    DECLARE @DealerDiscount money;
    SET @DealerDiscount = 0.60 -- 60% of list price
    SELECT @DealerPrice = plph.[ListPrice] * @DealerDiscount
    FROM [Production].[Product] p
        INNER JOIN [Production].[ProductListPriceHistory] plph
        ON p.[ProductID] = plph.[ProductID]
        AND p.[ProductID] = @ProductID
        AND @OrderDate BETWEEN plph.[StartDate] AND COALESCE(plph.[EndDate], CONVERT(datetime, '99991231', 112)); -- Make
        sure we get all the prices!
    RETURN @DealerPrice;
END;
```

#### 8.4.4.3. Function: dbo.ufnGetProductListPrice

Scalar function returning the list price for a given product on a particular order date.

##### Input/Output

	Name	Data type	Description
*@	Returns	money	
@	ProductID	int	Input parameter for the scalar function ufnGetProductListPrice. Enter a valid ProductID from the Production.Product table.
@	OrderDate	datetime	Input parameter for the scalar function ufnGetProductListPrice. Enter a valid order date.

##### Uses

Name
dbo.ufnGetProductListPrice
Production.Product
Production.ProductListPriceHistory

##### Script

```
CREATE FUNCTION [dbo].[ufnGetProductListPrice] (@ProductID [int], @OrderDate [datetime])
RETURNS [money]
AS
BEGIN
    DECLARE @ListPrice money;
    SELECT @ListPrice = plph.[ListPrice]
    FROM [Production].[Product] p
        INNER JOIN [Production].[ProductListPriceHistory] plph
        ON p.[ProductID] = plph.[ProductID]
        AND p.[ProductID] = @ProductID
        AND @OrderDate BETWEEN plph.[StartDate] AND COALESCE(plph.[EndDate], CONVERT(datetime, '99991231', 112)); -- Make
    sure we get all the prices!
    RETURN @ListPrice;
END;
```

#### 8.4.4.4. Function: dbo.ufnGetProductStandardCost

Scalar function returning the standard cost for a given product on a particular order date.

##### Input/Output

	Name	Data type	Description
*@	Returns	money	
*@	ProductID	int	Input parameter for the scalar function ufnGetProductStandardCost. Enter a valid ProductID from the Production.Product table.
*@	OrderDate	datetime	Input parameter for the scalar function ufnGetProductStandardCost. Enter a valid order date.

##### Uses

Name
dbo.ufnGetProductStandardCost
Production.Product
Production.ProductCostHistory

##### Script

```
CREATE FUNCTION [dbo].[ufnGetProductStandardCost] (@ProductID [int], @OrderDate [datetime])
RETURNS [money]
AS
-- Returns the standard cost for the product on a specific date.
BEGIN
    DECLARE @StandardCost money;
    SELECT @StandardCost = pch.[StandardCost]
    FROM [Production].[Product] p
        INNER JOIN [Production].[ProductCostHistory] pch
        ON p.[ProductID] = pch.[ProductID]
        AND p.[ProductID] = @ProductID
        AND @OrderDate BETWEEN pch.[StartDate] AND COALESCE(pch.[EndDate], CONVERT(datetime, '99991231', 112)); -- Make
sure we get all the prices!
    RETURN @StandardCost;
END;
```

#### 8.4.4.5. Function: dbo.ufnGetStock

Scalar function returning the quantity of inventory in LocationID 6 (Miscellaneous Storage) for a specified ProductID.

##### Input/Output

	Name	Data type	Description
*@	Returns	int	
*@	ProductID	int	Input parameter for the scalar function ufnGetStock. Enter a valid ProductID from the Production.ProductInventory table.

##### Uses

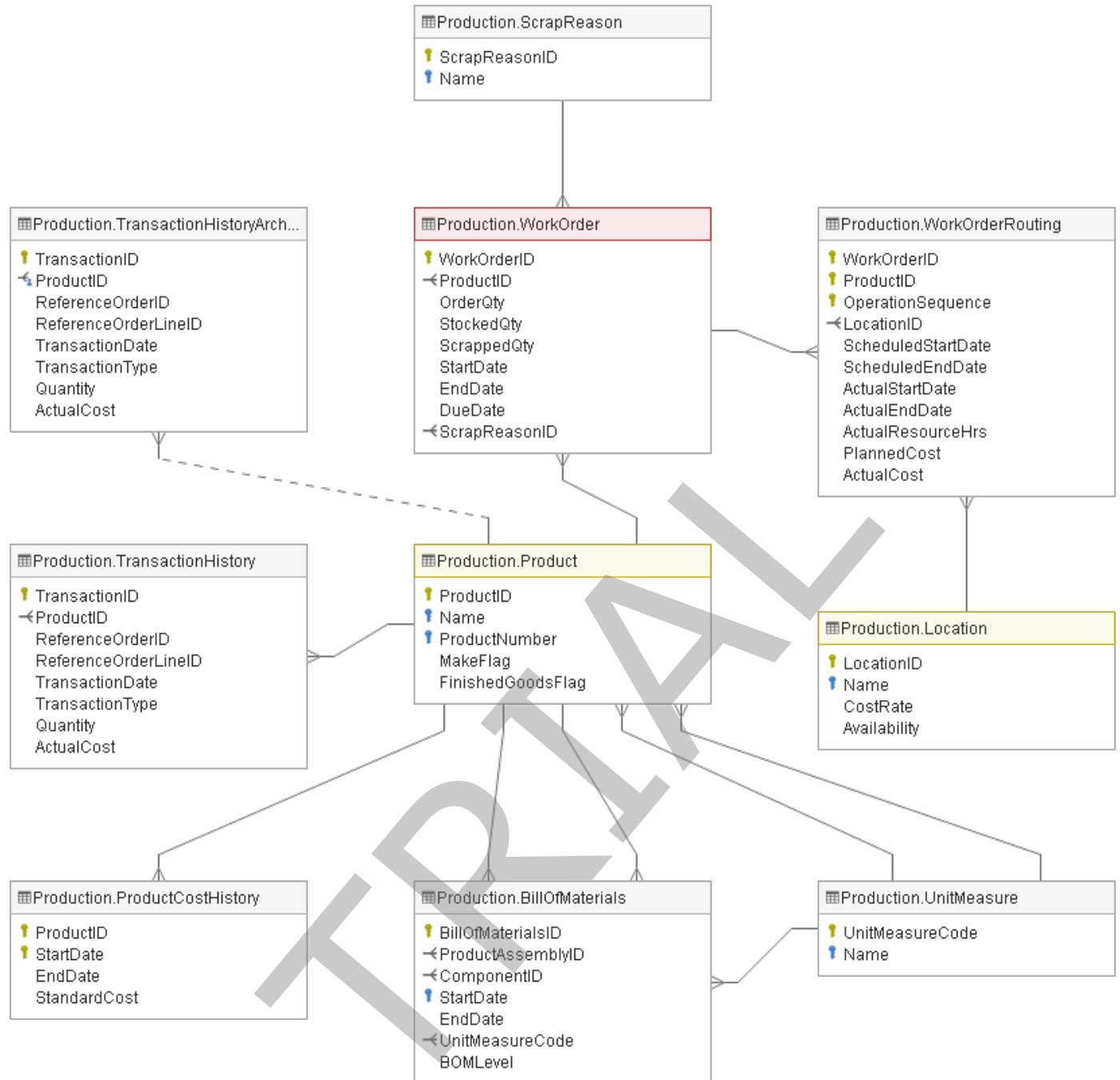
Name
dbo.ufnGetStock
Production.ProductInventory

##### Script

```
CREATE FUNCTION [dbo].[ufnGetStock] (@ProductID [int])
RETURNS [int]
AS
-- Returns the stock level for the product. This function is used internally only
BEGIN
    DECLARE @ret int;
    SELECT @ret = SUM(p.[Quantity])
    FROM [Production].[ProductInventory] p
    WHERE p.[ProductID] = @ProductID
        AND p.[LocationID] = '6'; -- Only look at inventory in the misc storage
    IF (@ret IS NULL)
        SET @ret = 0
    RETURN @ret
END;
```

TRIAL

## 8.5. Manufacturing



Manufacturing process management module.

## 8.5.1. Tables

### 8.5.1.1. Table: Production.BillOfMaterials

#### Sample field: Status: Active

Items required to make bicycles and bicycle subassemblies. It identifies the hierarchical relationship between a parent product and its components.

#### Columns

		Name	Data type	Description / Attributes
■	🔑	BillOfMaterialsID	int	Primary key for BillOfMaterials records. <b>Identity / Auto increment</b>
■	🔑	ProductAssemblyID	int	Parent product identification number. Foreign key to Product.ProductID. <b>Nullable</b> <b>References:</b> Production.Product
■	🔑	ComponentID	int	Component identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
■	🔑	StartDate	datetime	Date the component started being used in the assembly item. <b>Default:</b> getdate()
■		EndDate	datetime	Date the component stopped being used in the assembly item. <b>Nullable</b>
■		UnitMeasureCode	nchar(3)	Standard code identifying the unit of measure for the quantity. <b>References:</b> Production.UnitMeasure
■		BOMLevel	smallint	Indicates the depth the component is from its parent (AssemblyID).
■		PerAssemblyQty	decimal(8, 2)	Quantity of the component needed to create the assembly. <b>Default:</b> 1.00
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
→	Production.Product	<b>Production.BillOfMaterialsComponentID</b> = Production.ProductProductID	FK_BillOfMaterials_Product_ComponentID Foreign key constraint referencing Product.ProductAssemblyID.
→	Production.Product	<b>Production.BillOfMaterialsProductAssemblyID</b> = Production.ProductProductID	FK_BillOfMaterials_Product_ProductAssemblyID Foreign key constraint referencing Product.ProductAssemblyID.
→	Production.UnitMeasure	<b>Production.BillOfMaterialsUnitMeasureCode</b> = Production.UnitMeasureUnitMeasureCode	FK_BillOfMaterials_UnitMeasure_UnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.

#### Unique keys

Columns		Name / Description
🔑	BillOfMaterialsID	PK_BillOfMaterials_BillOfMaterialsID Primary key (clustered) constraint
🔑	ProductAssemblyID, ComponentID, StartDate	AK_BillOfMaterials_ProductAssemblyID_ComponentID_StartDate Clustered index.
🔑	BillOfMaterialsID	IX_BillOfMaterials
🔑	BillOfMaterialsID, ComponentID	UK_BillOfMaterials

## Uses

Name
Production.BillOfMaterials
→ Production.Product
→ Production.Product
→ Production.UnitMeasure

## Used By

Name
Production.BillOfMaterials
⚙️ dbo.uspGetBillOfMaterials
⚙️ dbo.uspGetWhereUsedProductID

TRIAL

## 8.5.1.2. Table: Production.ProductCostHistory

**Sample field: Status:** Active

Changes in the cost of a product over time.

### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductID	int	Product identification number. Foreign key to Product.ProductID <b>References:</b> Production.Product
■	🔑	StartDate	datetime	Product cost start date.
■		EndDate	datetime	Product cost end date. <b>Nullable</b>
■		StandardCost	money	Standard cost of the product.
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Product	Production.ProductCostHistory ProductID = Production.ProductProductID	FK_ProductCostHistory_Product_ProductID Foreign key constraint referencing Product.ProductID.

### Unique keys

Columns		Name / Description
🔑	ProductID, StartDate	PK_ProductCostHistory_ProductID_StartDate Primary key (clustered) constraint

### Uses

Name	
█	Production.ProductCostHistory
→	Production.Product

### Used By

Name	
█	Production.ProductCostHistory
fx	dbo.ufnGetProductStandardCost

### 8.5.1.3. Table: Production.ProductListPriceHistory

**Sample field: Status:** Active

Changes in the list price of a product over time.

#### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductID	int	Product identification number. Foreign key to Product.ProductID <b>References:</b> Production.Product
■	🔑	StartDate	datetime	List price start date.
■		EndDate	datetime	List price end date <b>Nullable</b>
■		ListPrice	money	Product list price.
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
→	Production.Product	Production.ProductListPriceHistoryProductID = Production.ProductProductID	FK_ProductListPriceHistory_Product_ProductID Foreign key constraint referencing Product.ProductID.

#### Unique keys

Columns		Name / Description
🔑	ProductID, StartDate	PK_ProductListPriceHistory_ProductID_StartDate Primary key (clustered) constraint

#### Uses

Name	
■	Production.ProductListPriceHistory
→	Production.Product

#### Used By

Name	
■	Production.ProductListPriceHistory
fx	dbo.ufnGetProductDealerPrice
fx	dbo.ufnGetProductListPrice

## 8.5.1.4. Table: Production.ScrapReason

**Sample field: Status:** Active

Manufacturing failure reasons lookup table.

### Columns

Name		Data type	Description / Attributes
ScrapReasonID		smallint	Primary key for ScrapReason records. <b>Identity / Auto increment</b>
Name		Name: nvarchar(50)	Failure description.
ModifiedDate		datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
Production.WorkOrder		Production.ScrapReasonScrapReasonID D = Production.WorkOrderScrapReasonID	FK_WorkOrder_ScrapReason_ScrapReasonID Foreign key constraint referencing ScrapReason.ScrapReasonID.

### Unique keys

Columns		Name / Description
	ScrapReasonID	PK_ScrapReason_ScrapReasonID Primary key (clustered) constraint
	Name	AK_ScrapReason_Name Unique nonclustered index.

### Used By

Name	
	Production.ScrapReason
	Production.WorkOrder

## 8.5.1.5. Table: Production.TransactionHistory

**Sample field: Status:** Active

Record of each purchase order, sales order, or work order transaction year to date.

### Columns

		Name	Data type	Description / Attributes
		TransactionID	int	Primary key for TransactionHistory records. <b>Identity / Auto increment</b>
		ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
		ReferenceOrderID	int	Purchase order, sales order, or work order identification number.
		ReferenceOrderLineID	int	Line number associated with the purchase order, sales order, or work order. <b>Default:</b> 0
		TransactionDate	datetime	Date and time of the transaction. <b>Default:</b> getdate()
		TransactionType	nchar(1)	W = WorkOrder, S = SalesOrder, P = PurchaseOrder
		Quantity	int	Product quantity.
		ActualCost	money	Product cost.
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table	Join	Title / Name / Description
Production.Product	Production.TransactionHistory Product D = Production.ProductProductID	FK_TransactionHistory_Product_ProductID Foreign key constraint referencing Product.ProductID.

### Unique keys

Columns	Name / Description
TransactionID	PK_TransactionHistory_TransactionID Primary key (clustered) constraint

### Uses

Name
Production.TransactionHistory
Production.Product

### Used By

Name
Production.TransactionHistory
Production.iWorkOrder
Production.uWorkOrder
Purchasing.iPurchaseOrderDetail
Purchasing.uPurchaseOrderDetail
Sales.iduSalesOrderDetail

## 8.5.1.6. Table: Production.TransactionHistoryArchive

**Sample field: Status:** Active

Transactions for previous years.

### Columns

	Name	Data type	Description / Attributes
■	TransactionID	int	Primary key for TransactionHistoryArchive records.
■	ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
■	ReferenceOrderID	int	Purchase order, sales order, or work order identification number.
■	ReferenceOrderLineID	int	Line number associated with the purchase order, sales order, or work order. <b>Default:</b> 0
■	TransactionDate	datetime	Date and time of the transaction. <b>Default:</b> getdate()
■	TransactionType	nchar(1)	W = Work Order, S = Sales Order, P = Purchase Order
■	Quantity	int	Product quantity.
■	ActualCost	money	Product cost.
■	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

	Table	Join	Title / Name / Description
→ ■	Production.Product	Production.TransactionHistoryArchive ProductID = Production.ProductProductID	User-defined relation

### Unique keys

	Columns	Name / Description
■	TransactionID	PK_TransactionHistoryArchive_TransactionID Primary key (clustered) constraint

### Uses

	Name
■	Production.TransactionHistoryArchive
→ ■	Production.Product

## 8.5.1.7. Table: Production.WorkOrder

**Sample field: Status:** Active

Manufacturing work orders.

### Columns

		Name	Data type	Description / Attributes
■	🔑	WorkOrderID	int	Primary key for WorkOrder records. <b>Identity / Auto increment</b>
■		ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
■		OrderQty	int	Product quantity to build.
■		StockedQty	int	Quantity built and put in inventory. <b>Computed:</b> (isnull([OrderQty]-[ScrappedQty],0))
■		ScrappedQty	smallint	Quantity that failed inspection.
■		StartDate	datetime	Work order start date.
■		EndDate	datetime	Work order end date. <b>Nullable</b>
■		DueDate	datetime	Work order due date.
■		ScrapReasonID	smallint	Reason for inspection failure. <b>Nullable</b> <b>References:</b> Production.ScrapReason
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Product	<b>Production.WorkOrder</b> ProductID = Production.ProductProductID	FK_WorkOrder_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Production.ScrapReason	<b>Production.WorkOrder</b> ScrapReasonID = Production.ScrapReasonScrapReasonID	FK_WorkOrder_ScrapReason_ScrapReasonID Foreign key constraint referencing ScrapReason.ScrapReasonID.

### Linked from

Table		Join	Title / Name / Description
←	Production.WorkOrderRouting	<b>Production.WorkOrder</b> WorkOrderID = Production.WorkOrderRoutingWorkOrderID	FK_WorkOrderRouting_WorkOrder_WorkOrderID Foreign key constraint referencing WorkOrder.WorkOrderID.

### Unique keys

Columns		Name / Description
🔑	WorkOrderID	PK_WorkOrder_WorkOrderID Primary key (clustered) constraint

## Triggers

	Name	When	Description
	iWorkOrder	After Insert	AFTER INSERT trigger that inserts a row in the TransactionHistory table.
			<pre> CREATE TRIGGER [Production].[iWorkOrder] ON [Production].[WorkOrder] AFTER INSERT AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         INSERT INTO [Production].[TransactionHistory] (             [ProductID]             ,[ReferenceOrderID]             ,[TransactionType]             ,[TransactionDate]             ,[Quantity]             ,[ActualCost])         SELECT             inserted.[ProductID]             ,inserted.[WorkOrderID]             ,'W'             ,GETDATE()             ,inserted.[OrderQty]             ,0         FROM inserted;     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[uspLogError];     END CATCH; END; </pre>

	Name	When	Description
	uWorkOrder	After Update	AFTER UPDATE trigger that inserts a row in the TransactionHistory table, updates ModifiedDate in the WorkOrder table.
			<pre> CREATE TRIGGER [Production].[uWorkOrder] ON [Production].[WorkOrder] AFTER UPDATE AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         IF UPDATE([ProductID]) OR UPDATE([OrderQty])         BEGIN             INSERT INTO [Production].[TransactionHistory] (                 [ProductID]                 ,[ReferenceOrderID]                 ,[TransactionType]                 ,[TransactionDate]                 ,[Quantity])             SELECT                 inserted.[ProductID]                 ,inserted.[WorkOrderID]                 ,'W'                 ,GETDATE()                 ,inserted.[OrderQty]             FROM inserted;         END;     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[uspLogError];     END CATCH; END; </pre>

## Uses

Name
Production.WorkOrder
Production.WorkOrder
Production.Product
Production.ScrapReason
Production.iWorkOrder
Production.TransactionHistory
dbo.uspLogError
dbo.uspPrintError
Production.uWorkOrder
Production.TransactionHistory
dbo.uspLogError
dbo.uspPrintError

## Used By

Name
Production.WorkOrder
Production.WorkOrder
Production.WorkOrderRouting

## 8.5.1.8. Table: Production.WorkOrderRouting

**Sample field: Status:** Active

Work order details.

### Columns

		Name	Data type	Description / Attributes
■	🔑	WorkOrderID	int	Primary key. Foreign key to WorkOrder.WorkOrderID. <b>References:</b> Production.WorkOrder
■	🔑	ProductID	int	Primary key. Foreign key to Product.ProductID.
■	🔑	OperationSequence	smallint	Primary key. Indicates the manufacturing process sequence.
■		LocationID	smallint	Manufacturing location where the part is processed. Foreign key to Location.LocationID. <b>References:</b> Production.Location
■		ScheduledStartDate	datetime	Planned manufacturing start date.
■		ScheduledEndDate	datetime	Planned manufacturing end date.
■		ActualStartDate	datetime	Actual start date. <b>Nullable</b>
■		ActualEndDate	datetime	Actual end date. <b>Nullable</b>
■		ActualResourceHrs	decimal(9, 4)	Number of manufacturing hours used. <b>Nullable</b>
■		PlannedCost	money	Estimated manufacturing cost.
■		ActualCost	money	Actual manufacturing cost. <b>Nullable</b>
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Location	Production.WorkOrderRoutingLocationID = Production.LocationLocationID	FK_WorkOrderRouting_Location_LocationID Foreign key constraint referencing Location.LocationID.
→	Production.WorkOrder	Production.WorkOrderRoutingWorkOrderID = Production.WorkOrderWorkOrderID	FK_WorkOrderRouting_WorkOrder_WorkOrderID Foreign key constraint referencing WorkOrder.WorkOrderID.

### Unique keys

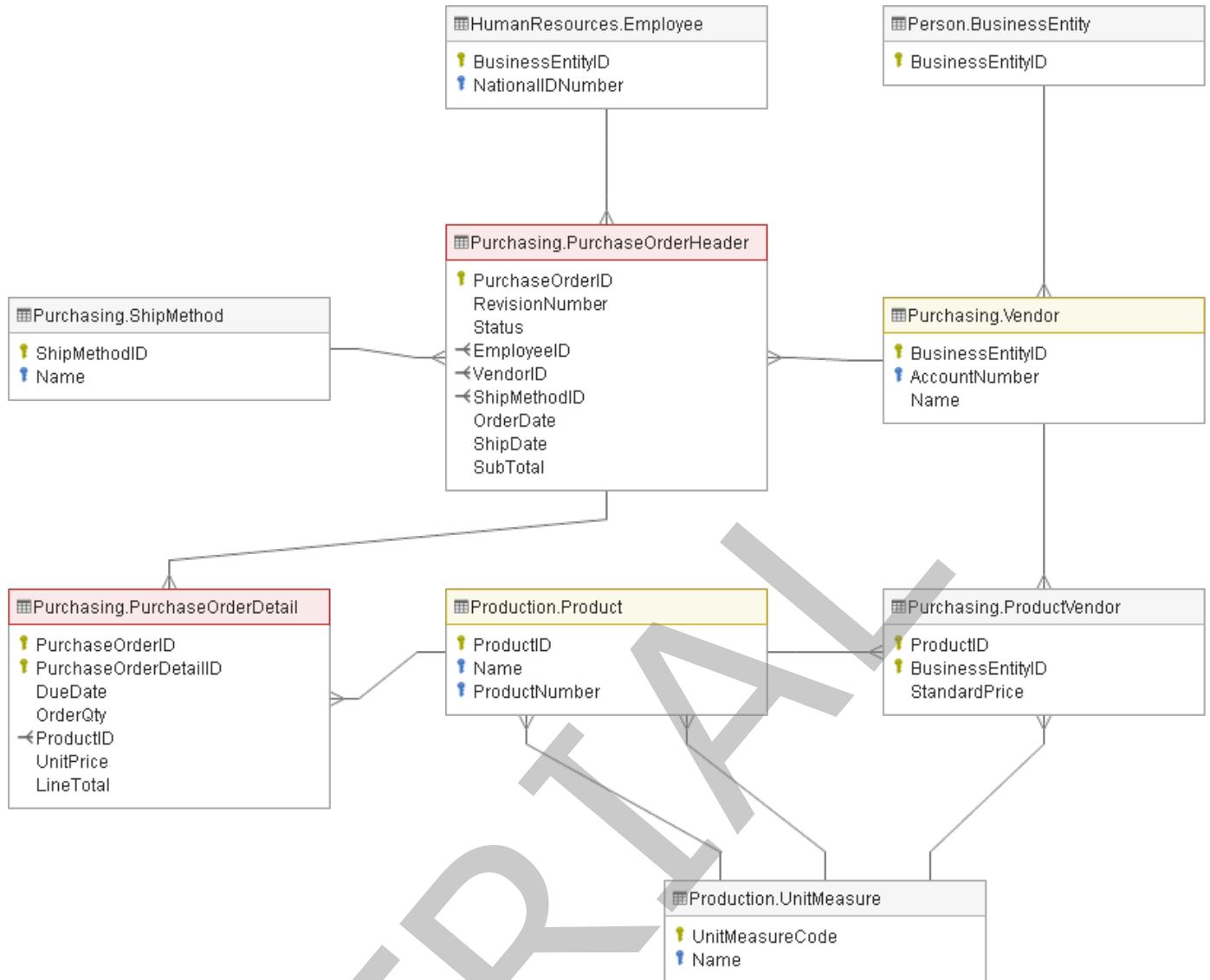
Columns		Name / Description
🔑	WorkOrderID, ProductID, OperationSequence	PK_WorkOrderRouting_WorkOrderID_ProductID_OperationSequence Primary key (clustered) constraint

### Uses

Name	
■	Production.WorkOrderRouting
→	Production.Location
→	Production.WorkOrder

TRIAL

## 8.6. Purchasing



Vendors from who parts and products are purchased.

## 8.6.1. Tables

### 8.6.1.1. Table: Purchasing.ProductVendor

**Sample field: Status:** Active

Cross-reference table mapping vendors with the products they supply.

#### Columns

	Name	Data type	Description / Attributes
☰	ProductID 	int	Primary key. Foreign key to Product.ProductID. <b>References:</b> Production.Product
☰	BusinessEntityID 	int	Primary key. Foreign key to Vendor.BusinessEntityID. <b>References:</b> Purchasing.Vendor
☰	AverageLeadTime	int	The average span of time (in days) between placing an order with the vendor and receiving the purchased product.
☰	StandardPrice	money	The vendor's usual selling price.
☰	LastReceiptCost	money	The selling price when last purchased. <b>Nullable</b>
☰	LastReceiptDate	datetime	Date the product was last received by the vendor. <b>Nullable</b>
☰	MinOrderQty	int	The maximum quantity that should be ordered.
☰	MaxOrderQty	int	The minimum quantity that should be ordered.
☰	OnOrderQty	int	The quantity currently on order. <b>Nullable</b>
☰	UnitMeasureCode	nchar(3)	The product's unit of measure. <b>References:</b> Production.UnitMeasure
☰	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

	Table	Join	Title / Name / Description
→	Production.Product	<b>Purchasing.ProductVendor</b> ProductID = Production.ProductProductID	FK_ProductVendor_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Production.UnitMeasure	<b>Purchasing.ProductVendor</b> UnitMeasureCode = Production.UnitMeasureUnitMeasureCode	FK_ProductVendor_UnitMeasure_UnitMeasureCode Foreign key constraint referencing UnitMeasure.UnitMeasureCode.
→	Purchasing.Vendor	<b>Purchasing.ProductVendor</b> BusinessEntityID = Purchasing.VendorBusinessEntityID	FK_ProductVendor_Vendor_BusinessEntityID Foreign key constraint referencing Vendor.BusinessEntityID.

#### Unique keys

	Columns	Name / Description
	ProductID, BusinessEntityID	PK_ProductVendor_ProductID_BusinessEntityID Primary key (clustered) constraint

## Uses

Name
Purchasing.ProductVendor
Production.Product
Production.UnitMeasure
Purchasing.Vendor

TRIAL

## 8.6.1.2. Table: Purchasing.PurchaseOrderDetail

**Sample field: Status:** Active

Individual products associated with a specific purchase order. See PurchaseOrderHeader.

### Columns

		Name	Data type	Description / Attributes
■	🔑	PurchaseOrderID	int	Primary key. Foreign key to PurchaseOrderHeader.PurchaseOrderID. <b>References:</b> Purchasing.PurchaseOrderHeader
■	🔑	PurchaseOrderDetailID	int	Primary key. One line number per purchased product. <b>Identity / Auto increment</b>
■		DueDate	datetime	Date the product is expected to be received.
■		OrderQty	smallint	Quantity ordered.
■		ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
■		UnitPrice	money	Vendor's selling price of a single product.
■		LineTotal	money	Per product subtotal. Computed as OrderQty * UnitPrice. <b>Computed:</b> (isnull([OrderQty]*[UnitPrice],(0.00)))
■		ReceivedQty	decimal(8, 2)	Quantity actually received from the vendor.
■		RejectedQty	decimal(8, 2)	Quantity rejected during inspection.
■		StockedQty	decimal(9, 2)	Quantity accepted into inventory. Computed as ReceivedQty - RejectedQty. <b>Computed:</b> (isnull([ReceivedQty]-[RejectedQty],(0.00)))
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Product	Purchasing.PurchaseOrderDetailProductID = Production.ProductProductID	FK_PurchaseOrderDetail_Product_ProductID Foreign key constraint referencing Product.ProductID.
→	Purchasing.PurchaseOrderHeader	Purchasing.PurchaseOrderDetailPurchaseOrderID = Purchasing.PurchaseOrderHeaderPurchaseOrderID	FK_PurchaseOrderDetail_PurchaseOrderHeader_PurchaseOrderID Foreign key constraint referencing PurchaseOrderHeader.PurchaseOrderID.

### Unique keys

Columns		Name / Description
🔑	PurchaseOrderID, PurchaseOrderDetailID	PK_PurchaseOrderDetail_PurchaseOrderID_PurchaseOrderDetailID Primary key (clustered) constraint

## Triggers

	Name	When	Description
	iPurchaseOrderDetail	After Insert	AFTER INSERT trigger that inserts a row in the TransactionHistory table and updates the PurchaseOrderHeader.SubTotal column.
<pre> CREATE TRIGGER [Purchasing].[iPurchaseOrderDetail] ON [Purchasing].[PurchaseOrderDetail] AFTER INSERT AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         INSERT INTO [Production].[TransactionHistory]         ([ProductID]         ,[ReferenceOrderID]         ,[ReferenceOrderLineID]         ,[TransactionType]         ,[TransactionDate]         ,[Quantity]         ,[ActualCost])         SELECT             inserted.[ProductID]             ,inserted.[PurchaseOrderID]             ,inserted.[PurchaseOrderDetailID]             ,'P'             ,GETDATE()             ,inserted.[OrderQty]             ,inserted.[UnitPrice]         FROM inserted         INNER JOIN [Purchasing].[PurchaseOrderHeader]         ON inserted.[PurchaseOrderID] = [Purchasing].[PurchaseOrderHeader].[PurchaseOrderID];         -- Update SubTotal in PurchaseOrderHeader record. Note that this causes the         -- PurchaseOrderHeader trigger to fire which will update the RevisionNumber.         UPDATE [Purchasing].[PurchaseOrderHeader]         SET [Purchasing].[PurchaseOrderHeader].[SubTotal] =         (SELECT SUM([Purchasing].[PurchaseOrderDetail].[LineTotal])         FROM [Purchasing].[PurchaseOrderDetail]         WHERE [Purchasing].[PurchaseOrderHeader].[PurchaseOrderID] =         [Purchasing].[PurchaseOrderDetail].[PurchaseOrderID])         WHERE [Purchasing].[PurchaseOrderHeader].[PurchaseOrderID] IN (SELECT inserted.[PurchaseOrderID] FROM inserted);     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[uspLogError];     END CATCH; END; </pre>			
	uPurchaseOrderDetail	After Update	AFTER UPDATE trigger that inserts a row in the TransactionHistory table, updates ModifiedDate in PurchaseOrderDetail and updates the PurchaseOrderHeader.SubTotal column.

Name	When	Description
<pre> CREATE TRIGGER [Purchasing].[uPurchaseOrderDetail] ON [Purchasing].[PurchaseOrderDetail] AFTER UPDATE AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         IF UPDATE([ProductID]) OR UPDATE([OrderQty]) OR UPDATE([UnitPrice])         -- Insert record into TransactionHistory         BEGIN             INSERT INTO [Production].[TransactionHistory]             ([ProductID]             ,[ReferenceOrderID]             ,[ReferenceOrderLineID]             ,[TransactionType]             ,[TransactionDate]             ,[Quantity]             ,[ActualCost])             SELECT                 inserted.[ProductID]                 ,inserted.[PurchaseOrderID]                 ,inserted.[PurchaseOrderDetailID]                 ,'P'                 ,GETDATE()                 ,inserted.[OrderQty]                 ,inserted.[UnitPrice]             FROM inserted             INNER JOIN [Purchasing].[PurchaseOrderDetail]             ON inserted.[PurchaseOrderID] = [Purchasing].[PurchaseOrderDetail].[PurchaseOrderID];             -- Update SubTotal in PurchaseOrderHeader record. Note that this causes the             -- PurchaseOrderHeader trigger to fire which will update the RevisionNumber.             UPDATE [Purchasing].[PurchaseOrderHeader]             SET [Purchasing].[PurchaseOrderHeader].[SubTotal] =             (SELECT SUM([Purchasing].[PurchaseOrderDetail].[LineTotal])             FROM [Purchasing].[PurchaseOrderDetail]             WHERE [Purchasing].[PurchaseOrderHeader].[PurchaseOrderID]             = [Purchasing].[PurchaseOrderDetail].[PurchaseOrderID])             WHERE [Purchasing].[PurchaseOrderHeader].[PurchaseOrderID]             IN (SELECT inserted.[PurchaseOrderID] FROM inserted);             UPDATE [Purchasing].[PurchaseOrderDetail]             SET [Purchasing].[PurchaseOrderDetail].[ModifiedDate] = GETDATE()             FROM inserted             WHERE inserted.[PurchaseOrderID] = [Purchasing].[PurchaseOrderDetail].[PurchaseOrderID]             AND inserted.[PurchaseOrderDetailID] = [Purchasing].[PurchaseOrderDetail].[PurchaseOrderDetailID];         END;     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[usp.LogError];     END CATCH; END; </pre>		

## Uses

Name
█ Purchasing.PurchaseOrderDetail
█ Purchasing.PurchaseOrderDetail
→ Production.Product
→ Purchasing.PurchaseOrderHeader
⚡ Purchasing.iPurchaseOrderDetail
█ Production.TransactionHistory
█ Purchasing.PurchaseOrderDetail
█ Purchasing.PurchaseOrderHeader
⚙ dbo.usp.LogError
⚙ dbo.uspPrintError

## Name

 Purchasing.uPurchaseOrderDetail

 Production.TransactionHistory

 Purchasing.PurchaseOrderDetail

 Purchasing.PurchaseOrderHeader

 dbo.uspLogError

 dbo.uspPrintError

## Used By

## Name

 Purchasing.PurchaseOrderDetail

 Purchasing.PurchaseOrderDetail

 Purchasing.iPurchaseOrderDetail

 Purchasing.uPurchaseOrderDetail

TRIAL

### 8.6.1.3. Table: Purchasing.PurchaseOrderHeader

**Sample field: Status:** Active

General purchase order information. See PurchaseOrderDetail.

#### Columns

		Name	Data type	Description / Attributes
	PurchaseOrderID		int	Primary key. <b>Identity / Auto increment</b>
	RevisionNumber		tinyint	Incremental number to track changes to the purchase order over time. <b>Default:</b> 0
	Status		tinyint	Order current status. 1 = Pending; 2 = Approved; 3 = Rejected; 4 = Complete <b>Default:</b> 1
	EmployeeID		int	Employee who created the purchase order. Foreign key to Employee.BusinessEntityID. <b>References:</b> HumanResources.Employee
	VendorID		int	Vendor with whom the purchase order is placed. Foreign key to Vendor.BusinessEntityID. <b>References:</b> Purchasing.Vendor
	ShipMethodID		int	Shipping method. Foreign key to ShipMethod.ShipMethodID. <b>References:</b> Purchasing.ShipMethod
	OrderDate		datetime	Purchase order creation date. <b>Default:</b> getdate()
	ShipDate		datetime	Estimated shipment date from the vendor. <b>Nullable</b>
	SubTotal		money	Purchase order subtotal. Computed as SUM(PurchaseOrderDetail.LineTotal)for the appropriate PurchaseOrderID. <b>Default:</b> 0.00
	TaxAmt		money	Tax amount. <b>Default:</b> 0.00
	Freight		money	Shipping cost. <b>Default:</b> 0.00
	TotalDue		money	Total due to vendor. Computed as Subtotal + TaxAmt + Freight. <b>Computed:</b> (isnull(([SubTotal]+[TaxAmt])+[Freight],(0)))
	ModifiedDate		datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
	HumanResources.Employee	<b>Purchasing.PurchaseOrderHeader</b> EmployeeID = HumanResources.EmployeeBusinessEntityID	FK_PurchaseOrderHeader_Employee_EmployeeID Foreign key constraint referencing Employee.EmployeeID.
	Purchasing.ShipMethod	<b>Purchasing.PurchaseOrderHeader</b> ShipMethodID = Purchasing.ShipMethodShipMethodID	FK_PurchaseOrderHeader_ShipMethod_ShipMethodID Foreign key constraint referencing ShipMethod.ShipMethodID.
	Purchasing.Vendor	<b>Purchasing.PurchaseOrderHeader</b> VendorID = Purchasing.VendorBusinessEntityID	FK_PurchaseOrderHeader_Vendor_VendorID Foreign key constraint referencing Vendor.VendorID.

## Linked from

	Table	Join	Title / Name / Description
→	Purchasing.PurchaseOrderDetail	<b>Purchasing.PurchaseOrderHeaderPurchaseOrderID = Purchasing.PurchaseOrderDetailPurchaseOrderID</b>	FK_PurchaseOrderDetail_PurchaseOrderHeader_PurchaseOrderID Foreign key constraint referencing PurchaseOrderHeader.PurchaseOrderID.

## Unique keys

	Columns	Name / Description
🔑	PurchaseOrderID	PK_PurchaseOrderHeader_PurchaseOrderID Primary key (clustered) constraint

## Triggers

	Name	When	Description
⚡	uPurchaseOrderHeader	After Update	AFTER UPDATE trigger that updates the RevisionNumber and ModifiedDate columns in the PurchaseOrderHeader table.
<pre>CREATE TRIGGER [Purchasing].[uPurchaseOrderHeader] ON [Purchasing].[PurchaseOrderHeader] AFTER UPDATE AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         -- Update RevisionNumber for modification of any field EXCEPT the Status.         IF NOT UPDATE([Status])             BEGIN                 UPDATE [Purchasing].[PurchaseOrderHeader]                 SET [Purchasing].[PurchaseOrderHeader].[RevisionNumber] =                     [Purchasing].[PurchaseOrderHeader].[RevisionNumber] + 1                 WHERE [Purchasing].[PurchaseOrderHeader].[PurchaseOrderID] IN                     (SELECT inserted.[PurchaseOrderID] FROM inserted);             END;     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[uspLogError];     END CATCH; END;</pre>			

## Uses

	Name
█	Purchasing.PurchaseOrderHeader
↳	Purchasing.PurchaseOrderHeader
→	HumanResources.Employee
→	Purchasing.ShipMethod
→	Purchasing.Vendor
⚡	Purchasing.uPurchaseOrderHeader
↳	Purchasing.PurchaseOrderHeader
⚙️	dbo.uspLogError
⚙️	dbo.uspPrintError

Used By

Name
■ Purchasing.PurchaseOrderHeader
■ Purchasing.PurchaseOrderHeader
⚡ Purchasing.iPurchaseOrderDetail
⚡ Purchasing.uPurchaseOrderDetail
⚡ Purchasing.uPurchaseOrderHeader
← Purchasing.PurchaseOrderDetail

TRIAL

## 8.6.1.4. Table: Purchasing.ShipMethod

**Sample field: Status:** Active

Shipping company lookup table.

### Columns

		Name	Data type	Description / Attributes
grid icon	key icon	ShipMethodID	int	Primary key for ShipMethod records. <b>Identity / Auto increment</b>
grid icon	key icon	Name	Name: nvarchar(50)	Shipping company name.
grid icon		ShipBase	money	Minimum shipping charge. <b>Default:</b> 0.00
grid icon		ShipRate	money	Shipping charge per pound. <b>Default:</b> 0.00
grid icon	key icon	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
grid icon		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

		Table	Join	Title / Name / Description
→	Purchasing.PurchaseOrderHeader		<b>Purchasing.ShipMethod</b> ShipMethodID = Purchasing.PurchaseOrderHeaderShipMethodID	FK_PurchaseOrderHeader_ShipMethod_ShipMethodID Foreign key constraint referencing ShipMethod.ShipMethodID.
→	Sales.SalesOrderHeader		<b>Purchasing.ShipMethod</b> ShipMethodID = Sales.SalesOrderHeaderShipMethodID	FK_SalesOrderHeader_ShipMethod_ShipMethodID Foreign key constraint referencing ShipMethod.ShipMethodID.

### Unique keys

		Columns	Name / Description
key icon	ShipMethodID	PK_ShipMethod_ShipMethodID	Primary key (clustered) constraint
key icon	Name	AK_ShipMethod_Name	Unique nonclustered index.
key icon	rowguid	AK_ShipMethod_rowguid	Unique nonclustered index. Used to support replication samples.

### Used By

		Name
grid icon	Purchasing.ShipMethod	
→	Purchasing.PurchaseOrderHeader	
→	Sales.SalesOrderHeader	

## 8.6.1.5. Table: Purchasing.Vendor

**Sample field: Status:** Active

Companies from whom Adventure Works Cycles purchases parts or other goods.

### Columns

		Name	Data type	Description / Attributes
■	🔑	BusinessEntityID	int	Primary key for Vendor records. Foreign key to BusinessEntity.BusinessEntityID <b>References:</b> Person.BusinessEntity
■	🔑	AccountNumber	AccountNumber: nvarchar(15)	Vendor account (identification) number.
■		Name	Name: nvarchar(50)	Company name.
■		CreditRating	tinyint	1 = Superior, 2 = Excellent, 3 = Above average, 4 = Average, 5 = Below average
■		PreferredVendorStatus	Flag: bit	0 = Do not use if another vendor is available. 1 = Preferred over other vendors supplying the same product. <b>Default:</b> 1
■		ActiveFlag	Flag: bit	0 = Vendor no longer used. 1 = Vendor is actively used. <b>Default:</b> 1
■		PurchasingWebServiceURL	nvarchar(1024)	Vendor URL. <b>Nullable</b>
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.BusinessEntity	<b>Purchasing.Vendor</b> BusinessEntityID = Person.BusinessEntityBusinessEntityID	FK_Vendor_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID

### Linked from

Table		Join	Title / Name / Description
→	Purchasing.ProductVendor	<b>Purchasing.Vendor</b> BusinessEntityID = Purchasing.ProductVendorBusinessEntityID	FK_ProductVendor_Vendor_BusinessEntityID Foreign key constraint referencing Vendor.BusinessEntityID.
→	Purchasing.PurchaseOrderHeader	<b>Purchasing.Vendor</b> BusinessEntityID = Purchasing.PurchaseOrderHeaderVendorID	FK_PurchaseOrderHeader_Vendor_VendorID Foreign key constraint referencing Vendor.VendorID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID	PK_Vendor_BusinessEntityID Primary key (clustered) constraint
🔑	AccountNumber	AK_Vendor_AccountNumber Unique nonclustered index.

## Triggers

	Name	When	Description
	dVendor	Instead Of Delete	INSTEAD OF DELETE trigger which keeps Vendors from being deleted.
<pre> CREATE TRIGGER [Purchasing].[dVendor] ON [Purchasing].[Vendor] INSTEAD OF DELETE NOT FOR REPLICATION AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         DECLARE @DeleteCount int;         SELECT @DeleteCount = COUNT(*) FROM deleted;         IF @DeleteCount &gt; 0         BEGIN             RAISERROR                 (N'Vendors cannot be deleted. They can only be marked as not active.', -- Message                 10, -- Severity.                 1); -- State.             -- Rollback any active or uncommittable transactions             IF @@TRANCOUNT &gt; 0             BEGIN                 ROLLBACK TRANSACTION;             END         END;     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[uspLogError];     END CATCH; END; </pre>			

## Uses

	Name
	Purchasing.Vendor
→	Person.BusinessEntity
	Purchasing.dVendor
	dbo.uspLogError
	dbo.uspPrintError

## Used By

	Name
	Purchasing.Vendor
	Purchasing.vVendorWithAddresses
	Purchasing.vVendorWithContacts
	dbo.ufnGetContactInformation
←	Purchasing.ProductVendor
←	Purchasing.PurchaseOrderHeader

## 8.6.2. Views

### 8.6.2.1. View: Purchasing.vVendorWithAddresses

**Sample field: Status:** Active

Vendor (company) names and addresses .

#### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Name	Name: nvarchar(50)	
3	AddressType	Name: nvarchar(50)	
4	AddressLine1	nvarchar(60)	
5	AddressLine2	nvarchar(60)	Nullable
6	City	nvarchar(30)	
7	StateProvinceName	Name: nvarchar(50)	
8	PostalCode	nvarchar(15)	
9	CountryRegionName	Name: nvarchar(50)	

#### Uses

Name
Purchasing.vVendorWithAddresses
Person.Address
Person.AddressType
Person.BusinessEntityAddress
Person.CountryRegion
Person.StateProvince
Purchasing.Vendor

#### Script

```
CREATE VIEW [Purchasing].[vVendorWithAddresses] AS
SELECT
    v.[BusinessEntityID]
    ,v.[Name]
    ,at.[Name] AS [AddressType]
    ,a.[AddressLine1]
    ,a.[AddressLine2]
    ,a.[City]
    ,sp.[Name] AS [StateProvinceName]
    ,a.[PostalCode]
    ,cr.[Name] AS [CountryRegionName]
FROM [Purchasing].[Vendor] v
INNER JOIN [Person].[BusinessEntityAddress] bea
ON bea.[BusinessEntityID] = v.[BusinessEntityID]
INNER JOIN [Person].[Address] a
ON a.[AddressID] = bea.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
INNER JOIN [Person].[AddressType] at
ON at.[AddressTypeID] = bea.[AddressTypeID];
```

## 8.6.2.2. View: Purchasing.vVendorWithContacts

**Sample field: Status:** Active

Vendor (company) names and the names of vendor employees to contact.

Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Name	Name: nvarchar(50)	
3	ContactType	Name: nvarchar(50)	
4	Title	nvarchar(8)	<b>Nullable</b>
5	FirstName	Name: nvarchar(50)	
6	MiddleName	Name: nvarchar(50)	<b>Nullable</b>
7	LastName	Name: nvarchar(50)	
8	Suffix	nvarchar(10)	<b>Nullable</b>
9	PhoneNumber	Phone: nvarchar(25)	<b>Nullable</b>
10	PhoneNumberType	Name: nvarchar(50)	<b>Nullable</b>
11	EmailAddress	nvarchar(50)	<b>Nullable</b>
12	EmailPromotion	int	

Uses

	Name
1	Purchasing.vVendorWithContacts
2	Person.BusinessEntityContact
3	Person.ContactType
4	Person.EmailAddress
5	Person.Person
6	Person.PersonPhone
7	Person.PhoneNumberType
8	Purchasing.Vendor

## Script

```
CREATE VIEW [Purchasing].[vVendorWithContacts] AS
SELECT
    v.[BusinessEntityID]
    ,v.[Name]
    ,ct.[Name] AS [ContactType]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,pp.[PhoneNumber]
        ,pnt.[Name] AS [PhoneNumberType]
    ,ea.[EmailAddress]
    ,p.[EmailPromotion]
FROM [Purchasing].[Vendor] v
INNER JOIN [Person].[BusinessEntityContact] bec
ON bec.[BusinessEntityID] = v.[BusinessEntityID]
INNER JOIN [Person].ContactType ct
ON ct.[ContactTypeID] = bec.[ContactTypeID]
INNER JOIN [Person].[Person] p
ON p.[BusinessEntityID] = bec.[PersonID]
LEFT OUTER JOIN [Person].[EmailAddress] ea
ON ea.[BusinessEntityID] = p.[BusinessEntityID]
LEFT OUTER JOIN [Person].[PersonPhone] pp
ON pp.[BusinessEntityID] = p.[BusinessEntityID]
LEFT OUTER JOIN [Person].[PhoneNumberType] pnt
ON pnt.[PhoneNumberTypeID] = pp.[PhoneNumberTypeID];
```

TRIAL

## 8.6.3. Functions

### 8.6.3.1. Function: dbo.ufnGetPurchaseOrderStatusText

Scalar function returning the text representation of the Status column in the PurchaseOrderHeader table.

#### Input/Output

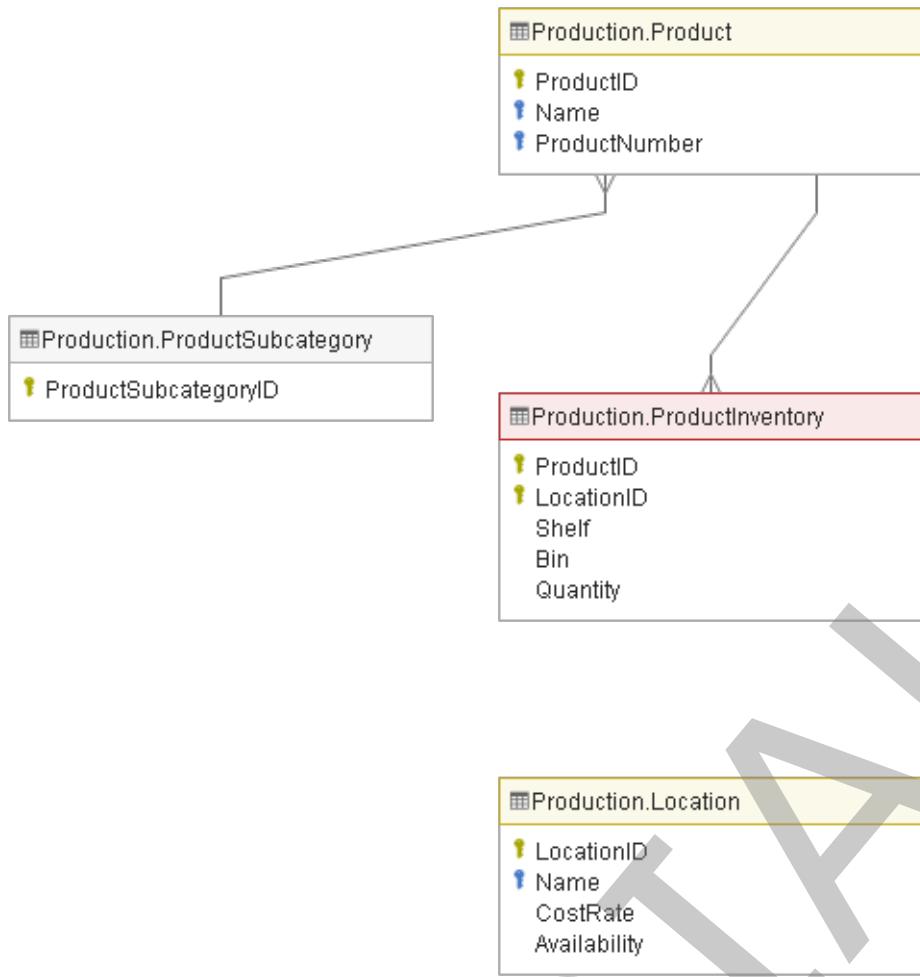
	Name	Data type	Description
*@	Returns	nvarchar(15)	
*@	Status	tinyint	Input parameter for the scalar function ufnGetPurchaseOrderStatusText. Enter a valid integer.

#### Script

```
CREATE FUNCTION [dbo].[ufnGetPurchaseOrderStatusText] (@Status [tinyint])
RETURNS [nvarchar] (15)
AS
-- Returns the sales order status text representation for the status value.
BEGIN
    DECLARE @ret [nvarchar] (15);
    SET @ret =
        CASE @Status
            WHEN 1 THEN 'Pending'
            WHEN 2 THEN 'Approved'
            WHEN 3 THEN 'Rejected'
            WHEN 4 THEN 'Complete'
            ELSE '** Invalid **'
        END;
    RETURN @ret
END;
```

TRIAL

## 8.7. Inventory



Inventory management.

## 8.7.1. Tables

### 8.7.1.1. Table: Production.Location

**Sample field: Status:** Active

Product inventory and manufacturing locations.

#### Columns

		Name	Data type	Description / Attributes
grid	key	LocationID	smallint	Primary key for Location records. <b>Identity / Auto increment</b>
grid	key	Name	Name: nvarchar(50)	Location description.
grid		CostRate	smallmoney	Standard hourly cost of the manufacturing location. <b>Default:</b> 0.00
grid		Availability	decimal(8, 2)	Work capacity (in hours) of the manufacturing location. <b>Default:</b> 0.00
grid		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

	Table	Join	Title / Name / Description
→	Production.ProductInventory	<b>Production.Location</b> LocationID = Production.ProductInventoryLocationID	FK_ProductInventory_Location_LocationID Foreign key constraint referencing Location.LocationID.
→	Production.WorkOrderRouting	<b>Production.Location</b> LocationID = Production.WorkOrderRoutingLocationID	FK_WorkOrderRouting_Location_LocationID Foreign key constraint referencing Location.LocationID.

#### Unique keys

	Columns	Name / Description
key	LocationID	PK_Location_LocationID Primary key (clustered) constraint
key	Name	AK_Location_Name Unique nonclustered index.

#### Used By

	Name
grid	<b>Production.Location</b>
→	Production.ProductInventory
→	Production.WorkOrderRouting

## 8.7.1.2. Table: Production.ProductInventory

**Sample field: Status:** Active

Product inventory information.

### Columns

		Name	Data type	Description / Attributes
■	🔑	ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
■	🔑	LocationID	smallint	Inventory location identification number. Foreign key to Location.LocationID. <b>References:</b> Production.Location
■		Shelf	nvarchar(10)	Storage compartment within an inventory location.
■		Bin	tinyint	Storage container on a shelf in an inventory location.
■		Quantity	smallint	Quantity of products in the inventory location. <b>Default:</b> 0
■		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Location	<b>Production.ProductInventory</b> LocationID = Production.Location.LocationID	FK_ProductInventory_Location_LocationID Foreign key constraint referencing Location.LocationID.
→	Production.Product	<b>Production.ProductInventory</b> ProductID = Production.Product.ProductID	FK_ProductInventory_Product_ProductID Foreign key constraint referencing Product.ProductID.

### Unique keys

Columns		Name / Description
🔑	ProductID, LocationID	<b>PK_ProductInventory_ProductID_LocationID</b> Primary key (clustered) constraint

### Uses

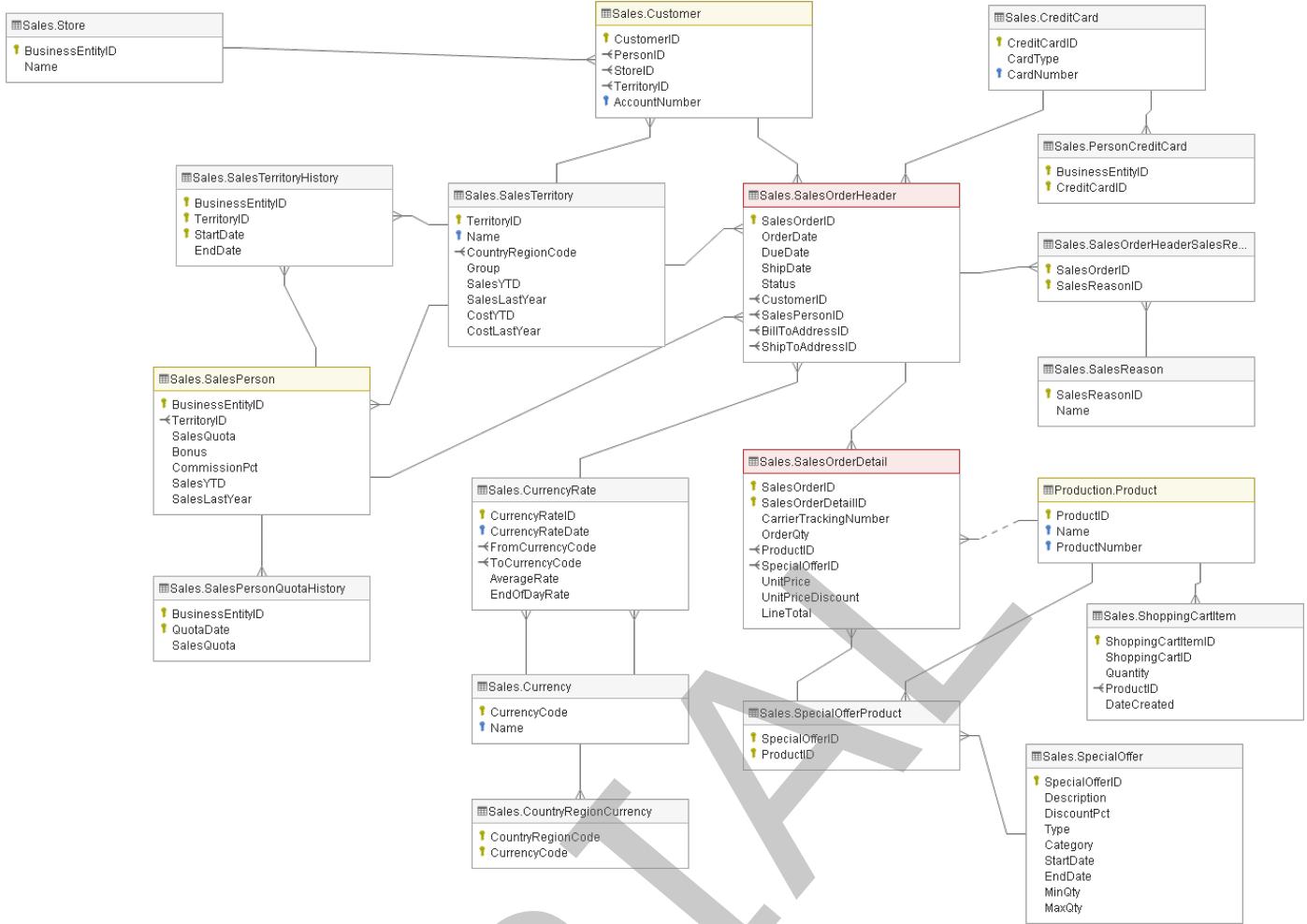
Name	
■	Production.ProductInventory
→	Production.Location
→	Production.Product

### Used By

Name	
■	Production.ProductInventory
‣	dbo.ufnGetStock

TRIAL

## 8.8. Sales



Sales module holds information about shopping cart, sales orders, special offers and sales people.

## 8.8.1. Tables

### 8.8.1.1. Table: Sales.CountryRegionCurrency

**Sample field: Status:** Active

Cross-reference table mapping ISO currency codes to a country or region.

#### Columns

		Name	Data type	Description / Attributes
		CountryRegionCode	nvarchar(3)	ISO code for countries and regions. Foreign key to CountryRegion.CountryRegionCode. <b>References:</b> Person.CountryRegion
		CurrencyCode	nchar(3)	ISO standard currency code. Foreign key to Currency.CurrencyCode. <b>References:</b> Sales.Currency
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
	Person.CountryRegion	Sales.CountryRegionCurrencyCountryRegionCode = Person.CountryRegionCountryRegionCode	FK_CountryRegionCurrency_CountryRegion_CountryRegionCode Foreign key constraint referencing CountryRegion.CountryRegionCode.
	Sales.Currency	Sales.CountryRegionCurrencyCurrencyCode = Sales.CurrencyCurrencyCode	FK_CountryRegionCurrency_Currency_CurrencyCode Foreign key constraint referencing Currency.CurrencyCode.

#### Unique keys

Columns		Name / Description
	CountryRegionCode, CurrencyCode	PK_CountryRegionCurrency_CountryRegionCode_CurrencyCode Primary key (clustered) constraint

#### Uses

Name	
	Sales.CountryRegionCurrency
	Person.CountryRegion
	Sales.Currency

## 8.8.1.2. Table: Sales.CreditCard

**Sample field: Status:** Active

Customer credit card information.

### Columns

		Name	Data type	Description / Attributes
█	🔑	CreditCardID	int	Primary key for CreditCard records. <b>Identity / Auto increment</b>
█		CardType	nvarchar(50)	Credit card name.
█	🔑	CardNumber	nvarchar(25)	Credit card number.
█		ExpMonth	tinyint	Credit card expiration month.
█		ExpYear	smallint	Credit card expiration year.
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
→	Sales.PersonCreditCard	Sales.CreditCard.CreditCardID = Sales.PersonCreditCard.CreditCardID	FK_PersonCreditCard_CreditCard_CreditCardID Foreign key constraint referencing CreditCard.CreditCardID.
→	Sales.SalesOrderHeader	Sales.CreditCard.CreditCardID = Sales.SalesOrderHeader.CreditCardID	FK_SalesOrderHeader_CreditCard_CreditCardID Foreign key constraint referencing CreditCard.CreditCardID.

### Unique keys

Columns		Name / Description
🔑	CreditCardID	PK_CreditCard_CreditCardID Primary key (clustered) constraint
🔑	CardNumber	AK_CreditCard_CardNumber Unique nonclustered index.

### Used By

Name	
█	Sales.CreditCard
→	Sales.PersonCreditCard
→	Sales.SalesOrderHeader

### 8.8.1.3. Table: Sales.Currency

**Sample field: Status:** Active

Lookup table containing standard ISO currencies.

#### Columns

		Name	Data type	Description / Attributes
		CurrencyCode	nchar(3)	The ISO code for the Currency.
		Name	Name: nvarchar(50)	Currency name.
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Linked from

Table		Join	Title / Name / Description
→	Sales.CountryRegionCurrency	<b>Sales.Currency</b> CurrencyCode = Sales.CountryRegionCurrencyCurrencyCode	FK_CountryRegionCurrency_Currency_CurrencyCode Foreign key constraint referencing Currency.CurrencyCode.
→	Sales.CurrencyRate	<b>Sales.Currency</b> CurrencyCode = Sales.CurrencyRateFromCurrencyCode	FK_CurrencyRate_Currency_FromCurrencyCode Foreign key constraint referencing Currency.FromCurrencyCode.
→	Sales.CurrencyRate	<b>Sales.Currency</b> CurrencyCode = Sales.CurrencyRateToCurrencyCode	FK_CurrencyRate_Currency_ToCurrencyCode Foreign key constraint referencing Currency.FromCurrencyCode.

#### Unique keys

Columns		Name / Description
	CurrencyCode	PK_Currency_CurrencyCode Primary key (clustered) constraint
	Name	AK_Currency_Name Unique nonclustered index.

#### Used By

Name	
	<b>Sales.Currency</b>
→	Sales.CountryRegionCurrency
→	Sales.CurrencyRate
→	Sales.CurrencyRate

## 8.8.1.4. Table: Sales.CurrencyRate

**Sample field: Status:** Active

Currency exchange rates.

### Columns

		Name	Data type	Description / Attributes
█	🔑	CurrencyRateID	int	Primary key for CurrencyRate records. <b>Identity / Auto increment</b>
█	🔑	CurrencyRateDate	datetime	Date and time the exchange rate was obtained.
█	🔑	FromCurrencyCode	nchar(3)	Exchange rate was converted from this currency code. <b>References:</b> Sales.Currency
█	🔑	ToCurrencyCode	nchar(3)	Exchange rate was converted to this currency code. <b>References:</b> Sales.Currency
█		AverageRate	money	Average exchange rate for the day.
█		EndOfDayRate	money	Final exchange rate for the day.
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

		Table	Join	Title / Name / Description
→	Sales.Currency		<b>Sales.CurrencyRateFromCurrencyCode</b> = Sales.CurrencyCurrencyCode	FK_CurrencyRate_Currency_FromCurrencyCode Foreign key constraint referencing Currency.FromCurrencyCode.
→	Sales.Currency		<b>Sales.CurrencyRateToCurrencyCode</b> = Sales.CurrencyCurrencyCode	FK_CurrencyRate_Currency_ToCurrencyCode Foreign key constraint referencing Currency.FromCurrencyCode.

### Linked from

		Table	Join	Title / Name / Description
←	Sales.SalesOrderHeader		<b>Sales.CurrencyRateCurrencyRateID</b> = Sales.SalesOrderHeaderCurrencyRateID	FK_SalesOrderHeader_CurrencyRate_CurrencyRateID Foreign key constraint referencing CurrencyRate.CurrencyRateID.

### Unique keys

		Columns	Name / Description
🔑	CurrencyRateID		PK_CurrencyRate_CurrencyRateID Primary key (clustered) constraint
🔑	CurrencyRateDate, FromCurrencyCode, ToCurrencyCode		AK_CurrencyRate_CurrencyRateDate_FromCurrencyCode_ToCurrencyCode Unique nonclustered index.

### Uses

		Name
█	<b>Sales.CurrencyRate</b>	
→	Sales.Currency	
→	Sales.Currency	

Used By

Name
Sales.CurrencyRate
← Sales.SalesOrderHeader

TRIAL

## 8.8.1.5. Table: Sales.Customer

### Sample field: Status: Active

Current customer information. Also see the Person and Store tables.

### Columns

		Name	Data type	Description / Attributes
■	🔑	CustomerID	int	Primary key. <b>Identity / Auto increment</b>
■		PersonID	int	Foreign key to Person.BusinessEntityID <b>Nullable</b> <b>References:</b> Person.Person
■		StoreID	int	Foreign key to Store.BusinessEntityID <b>Nullable</b> <b>References:</b> Sales.Store
■		TerritoryID	int	ID of the territory in which the customer is located. Foreign key to SalesTerritory.SalesTerritoryID. <b>Nullable</b> <b>References:</b> Sales.SalesTerritory
■	🔑	AccountNumber	varchar(10)	Unique number identifying the customer assigned by the accounting system. <b>Computed:</b> (isnull('AW'+[dbo].[ufnLeadingZeros]([CustomerID]),''))
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.Person	<b>Sales.Customer</b> PersonID = Person.PersonBusinessEntityID	FK_Customer_Person_PersonID Foreign key constraint referencing Person.BusinessEntityID.
→	Sales.SalesTerritory	<b>Sales.Customer</b> TerritoryID = Sales.SalesTerritoryTerritoryID	FK_Customer_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.
→	Sales.Store	<b>Sales.Customer</b> StoreID = Sales.StoreBusinessEntityID	FK_Customer_Store_StoreID Foreign key constraint referencing Store.BusinessEntityID.

### Linked from

Table		Join	Title / Name / Description
→	Sales.SalesOrderHeader	<b>Sales.Customer</b> CustomerID = Sales.SalesOrderHeaderCustomerID	FK_SalesOrderHeader_Customer_CustomerID Foreign key constraint referencing Customer.CustomerID.

### Unique keys

Columns		Name / Description
🔑	CustomerID	PK_Customer_CustomerID Primary key (clustered) constraint
🔑	AccountNumber	AK_Customer_AccountNumber Unique nonclustered index.
🔑	rowguid	AK_Customer_rowguid Unique nonclustered index. Used to support replication samples.

## Uses

Name
Sales.Customer
Sales.Customer
dbo.ufnLeadingZeros
Person.Person
Sales.SalesTerritory
Sales.Store

## Used By

Name
Sales.Customer
Sales.Customer
Sales.vlndividualCustomer
dbo.ufnGetContactInformation
Sales.iduSalesOrderDetail
Sales.SalesOrderHeader

## 8.8.1.6. Table: Sales.PersonCreditCard

**Sample field: Status:** Active

Cross-reference table mapping people to their credit card information in the CreditCard table.

### Columns

Name			Data type	Description / Attributes
BusinessEntityID		BusinessEntityID	int	Business entity identification number. Foreign key to Person.BusinessEntityID. <b>References:</b> Person.Person
CreditCardID		CreditCardID	int	Credit card identification number. Foreign key to CreditCard.CreditCardID. <b>References:</b> Sales.CreditCard
ModifiedDate		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

	Table	Join	Title / Name / Description
→	Sales.CreditCard	<b>Sales.PersonCreditCard</b> CreditCardID = Sales.CreditCardCreditCardID	FK_PersonCreditCard_CreditCard_CreditCardID Foreign key constraint referencing CreditCard.CreditCardID.
→	Person.Person	<b>Sales.PersonCreditCard</b> BusinessEntityID = Person.PersonBusinessEntityID	FK_PersonCreditCard_Person_BusinessEntityID Foreign key constraint referencing Person.BusinessEntityID.

### Unique keys

	Columns	Name / Description
	BusinessEntityID, CreditCardID	PK_PersonCreditCard_BusinessEntityID_CreditCardID Primary key (clustered) constraint

### Uses

	Name
	Sales.PersonCreditCard
→	Person.Person
→	Sales.CreditCard

## 8.8.1.7. Table: Sales.SalesOrderDetail

**Sample field: Status:** Active

Individual products associated with a specific sales order. See SalesOrderHeader.

### Columns

		Name	Data type	Description / Attributes
█	🔑	SalesOrderID	int	Primary key. Foreign key to SalesOrderHeader.SalesOrderID. <b>References:</b> Sales.SalesOrderHeader
█	🔑	SalesOrderDetailID	int	Primary key. One incremental unique number per product sold. <b>Identity / Auto increment</b>
█		CarrierTrackingNumber	nvarchar(25)	Shipment tracking number supplied by the shipper. <b>Nullable</b>
█		OrderQty	smallint	Quantity ordered per product.
█		ProductID	int	Product sold to customer. Foreign key to Product.ProductID. <b>References:</b> Production.Product, Sales.SpecialOfferProduct
█		SpecialOfferID	int	Promotional code. Foreign key to SpecialOffer.SpecialOfferID. <b>References:</b> Sales.SpecialOfferProduct
█		UnitPrice	money	Selling price of a single product.
█		UnitPriceDiscount	money	Discount amount. <b>Default:</b> 0.0
█		LineTotal	numeric(38, 6)	Per product subtotal. Computed as UnitPrice * (1 - UnitPriceDiscount) * OrderQty. <b>Computed:</b> (isnull(([UnitPrice]*(1.0)-[UnitPriceDiscount]))*[OrderQty],(0.0))
█	💡	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Production.Product	<b>Sales.SalesOrderDetail</b> ProductID = Production.ProductProductID	User-defined relation
→	Sales.SalesOrderHeader	<b>Sales.SalesOrderDetail</b> SalesOrderID = Sales.SalesOrderHeaderSalesOrderID	FK_SalesOrderDetail_SalesOrderHeader_SalesOrderID Foreign key constraint referencing SalesOrderHeader.PurchaseOrderID.
→	Sales.SpecialOfferProduct	<b>Sales.SalesOrderDetail</b> SpecialOfferID = Sales.SpecialOfferProductSpecialOfferID, <b>Sales.SalesOrderDetail</b> ProductID = Sales.SpecialOfferProductProductID	FK_SalesOrderDetail_SpecialOfferProduct_SpecialOfferIDProductID Foreign key constraint referencing SpecialOfferProduct.SpecialOfferIDProductID.

### Unique keys

Columns		Name / Description
🔑	SalesOrderID, SalesOrderDetailID	PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID Primary key (clustered) constraint
🔑	rowguid	AK_SalesOrderDetail_rowguid Unique nonclustered index. Used to support replication samples.

## Triggers

	Name	When	Description
	iduSalesOrderDetail	After Insert, Update, Delete	AFTER INSERT, DELETE, UPDATE trigger that inserts a row in the TransactionHistory table, updates ModifiedDate in SalesOrderDetail and updates the SalesOrderHeader.SubTotal column.
<pre> CREATE TRIGGER [Sales].[iduSalesOrderDetail] ON [Sales].[SalesOrderDetail] AFTER INSERT, DELETE, UPDATE AS BEGIN     DECLARE @Count int;     SET @Count = @@ROWCOUNT;     IF @Count = 0         RETURN;     SET NOCOUNT ON;     BEGIN TRY         -- If inserting or updating these columns         IF UPDATE([ProductID]) OR UPDATE([OrderQty]) OR UPDATE([UnitPrice]) OR UPDATE([UnitPriceDiscount])         -- Insert record into TransactionHistory         BEGIN             INSERT INTO [Production].[TransactionHistory]             ([ProductID]             ,[ReferenceOrderID]             ,[ReferenceOrderLineID]             ,[TransactionType]             ,[TransactionDate]             ,[Quantity]             ,[ActualCost])             SELECT                 inserted.[ProductID]                 ,inserted.[SalesOrderID]                 ,inserted.[SalesOrderDetailID]                 ,'I'                 ,GETDATE()                 ,inserted.[OrderQty]                 ,inserted.[UnitPrice]             FROM inserted             INNER JOIN [Sales].[SalesOrderHeader]             ON inserted.[SalesOrderID] = [Sales].[SalesOrderHeader].[SalesOrderID];             UPDATE [Person].[Person]             SET [Demographics].modify('declare default element namespace             "http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";             replace value of (/IndividualSurvey/TotalPurchaseYTD)[1]             with data(/IndividualSurvey/TotalPurchaseYTD)[1] + sql:column ("inserted.LineTotal")')             FROM inserted             INNER JOIN [Sales].[SalesOrderHeader] AS SOH             ON inserted.[SalesOrderID] = SOH.[SalesOrderID]             INNER JOIN [Sales].[Customer] AS C             ON SOH.[CustomerID] = C.[CustomerID]             WHERE C.[PersonID] = [Person].[Person].[BusinessEntityID];         END;         -- Update SubTotal in SalesOrderHeader record. Note that this causes the         -- SalesOrderHeader trigger to fire which will update the RevisionNumber.         UPDATE [Sales].[SalesOrderHeader]         SET [Sales].[SalesOrderHeader].[SubTotal] =         (SELECT SUM([Sales].[SalesOrderDetail].[LineTotal])         FROM [Sales].[SalesOrderDetail]         WHERE [Sales].[SalesOrderHeader].[SalesOrderID] = [Sales].[SalesOrderDetail].[SalesOrderID])         WHERE [Sales].[SalesOrderHeader].[SalesOrderID] IN (SELECT inserted.[SalesOrderID] FROM inserted);         UPDATE [Person].[Person]         SET [Demographics].modify('declare default element namespace         "http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";         replace value of (/IndividualSurvey/TotalPurchaseYTD)[1]         with data(/IndividualSurvey/TotalPurchaseYTD)[1] - sql:column("deleted.LineTotal")')         FROM deleted         INNER JOIN [Sales].[SalesOrderHeader]         ON deleted.[SalesOrderID] = [Sales].[SalesOrderHeader].[SalesOrderID]         INNER JOIN [Sales].[Customer]         ON [Sales].[Customer].[CustomerID] = [Sales].[SalesOrderHeader].[CustomerID]         WHERE [Sales].[Customer].[PersonID] = [Person].[Person].[BusinessEntityID];     END TRY     BEGIN CATCH         EXECUTE [dbo].[uspPrintError];         -- Rollback any active or uncommittable transactions before         -- inserting information in the ErrorLog         IF @@TRANCOUNT &gt; 0         BEGIN             ROLLBACK TRANSACTION;         END         EXECUTE [dbo].[uspLogError];     END CATCH; END; </pre>			

## Uses

Name
Sales.SalesOrderDetail
Sales.SalesOrderDetail
Production.Product
Sales.SalesOrderHeader
Sales.SpecialOfferProduct
Sales.iduSalesOrderDetail
Person.Person
Production.TransactionHistory
Sales.Customer
Sales.SalesOrderDetail
Sales.SalesOrderHeader
dbo.uspLogError
dbo.uspPrintError

## Used By

Name
Sales.SalesOrderDetail
Sales.SalesOrderDetail
Sales.iduSalesOrderDetail

### 8.8.1.8. Table: Sales.SalesOrderHeader

**Sample field: Status:** Active

General sales order information.

#### Columns

	Name	Data type	Description / Attributes
█	SalesOrderID	int	Primary key. <b>Identity / Auto increment</b>
█	RevisionNumber	tinyint	Incremental number to track changes to the sales order over time. <b>Default:</b> 0
█	OrderDate	datetime	Dates the sales order was created. <b>Default:</b> getdate()
█	DueDate	datetime	Date the order is due to the customer.
█	ShipDate	datetime	Date the order was shipped to the customer. <b>Nullable</b>
█	Status	tinyint	Order current status. 1 = In process; 2 = Approved; 3 = Backordered; 4 = Rejected; 5 = Shipped; 6 = Cancelled <b>Default:</b> 1
█	OnlineOrderFlag	Flag: bit	0 = Order placed by sales person. 1 = Order placed online by customer. <b>Default:</b> 1
█	SalesOrderNumber	nvarchar(25)	Unique sales order identification number. <b>Computed:</b> (isnull(N'SO'+CONVERT([nvarchar](23),[SalesOrderID]),N'*** ERROR ***'))
█	PurchaseOrderNumber	OrderNumber: nvarchar(25)	Customer purchase order number reference. <b>Nullable</b>
█	AccountNumber	AccountNumber: nvarchar(15)	Financial accounting number reference. <b>Nullable</b>
█	CustomerID	int	Customer identification number. Foreign key to Customer.BusinessEntityID. <b>References:</b> Sales.Customer
█	SalesPersonID	int	Sales person who created the sales order. Foreign key to SalesPerson.BusinessEntityID. <b>Nullable</b> <b>References:</b> Sales.SalesPerson
█	TerritoryID	int	Territory in which the sale was made. Foreign key to SalesTerritory.SalesTerritoryID. <b>Nullable</b> <b>References:</b> Sales.SalesTerritory
█	BillToAddressID	int	Customer billing address. Foreign key to Address.AddressID. <b>References:</b> Person.Address
█	ShipToAddressID	int	Customer shipping address. Foreign key to Address.AddressID. <b>References:</b> Person.Address
█	ShipMethodID	int	Shipping method. Foreign key to ShipMethod.ShipMethodID. <b>References:</b> Purchasing.ShipMethod
█	CreditCardID	int	Credit card identification number. Foreign key to CreditCard.CreditCardID. <b>Nullable</b> <b>References:</b> Sales.CreditCard
█	CreditCardApprovalCode	varchar(15)	Approval code provided by the credit card company. <b>Nullable</b>

Name		Data type	Description / Attributes
	CurrencyRateID	int	Currency exchange rate used. Foreign key to CurrencyRate.CurrencyRateID. <b>Nullable</b> <b>References:</b> Sales.CurrencyRate
	SubTotal	money	Sales subtotal. Computed as SUM(SalesOrderDetail.LineTotal)for the appropriate SalesOrderID. <b>Default:</b> 0.00
	TaxAmt	money	Tax amount. <b>Default:</b> 0.00
	Freight	money	Shipping cost. <b>Default:</b> 0.00
	TotalDue	money	Total due from customer. Computed as Subtotal + TaxAmt + Freight. <b>Computed:</b> (isnull(([SubTotal]+[TaxAmt])+[Freight],0))
	Comment	nvarchar(128)	Sales representative comments. <b>Nullable</b>
	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

## Links to

Table		Join	Title / Name / Description
→	Person.Address	<b>Sales.SalesOrderHeaderBillToAddressID</b> D = Person.AddressAddressID	FK_SalesOrderHeader_Address_BillToAddressID Foreign key constraint referencing Address.AddressID.
→	Person.Address	<b>Sales.SalesOrderHeaderShipToAddressID</b> sID = Person.AddressAddressID	FK_SalesOrderHeader_Address_ShipToAddressID Foreign key constraint referencing Address.AddressID.
→	Sales.CreditCard	<b>Sales.SalesOrderHeaderCreditCardID</b> = Sales.CreditCardCreditCardID	FK_SalesOrderHeader_CreditCard_CreditCardID Foreign key constraint referencing CreditCard.CreditCardID.
→	Sales.CurrencyRate	<b>Sales.SalesOrderHeaderCurrencyRateID</b> D = Sales.CurrencyRateCurrencyRateID	FK_SalesOrderHeader_CurrencyRate_CurrencyRateID Foreign key constraint referencing CurrencyRate.CurrencyRateID.
→	Sales.Customer	<b>Sales.SalesOrderHeaderCustomerID</b> = Sales.CustomerCustomerID	FK_SalesOrderHeader_Customer_CustomerID Foreign key constraint referencing Customer.CustomerID.
→	Sales.SalesPerson	<b>Sales.SalesOrderHeaderSalesPersonID</b> = Sales.SalesPersonBusinessEntityID	FK_SalesOrderHeader_SalesPerson_SalesPersonID Foreign key constraint referencing SalesPerson.SalesPersonID.
→	Sales.SalesTerritory	<b>Sales.SalesOrderHeaderTerritoryID</b> = Sales.SalesTerritoryTerritoryID	FK_SalesOrderHeader_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.
→	Purchasing.ShipMethod	<b>Sales.SalesOrderHeaderShipMethodID</b> = Purchasing.ShipMethodShipMethodID	FK_SalesOrderHeader_ShipMethod_ShipMethodID Foreign key constraint referencing ShipMethod.ShipMethodID.

## Linked from

Table		Join	Title / Name / Description
→	Sales.SalesOrderDetail	<b>Sales.SalesOrderHeaderSalesOrderID</b> = Sales.SalesOrderDetailSalesOrderID	FK_SalesOrderDetail_SalesOrderHeader_SalesOrderID Foreign key constraint referencing SalesOrderHeader.PurchaseOrderID.
→	Sales.SalesOrderHeaderSalesReason	<b>Sales.SalesOrderHeaderSalesOrderID</b> = Sales.SalesOrderHeaderSalesReasonSalesOrderID	FK_SalesOrderHeaderSalesReason_SalesOrderHeader_SalesOrderID Foreign key constraint referencing SalesOrderHeader.SalesOrderID.

## Unique keys

Columns		Name / Description
PK_SalesOrderHeader_SalesOrderID	SalesOrderID	Primary key (clustered) constraint
AK_SalesOrderHeader_rowguid	rowguid	Unique nonclustered index. Used to support replication samples.
AK_SalesOrderHeader_SalesOrderNumber	SalesOrderNumber	Unique nonclustered index.

## Triggers

	Name	When	Description
⚡	uSalesOrderHeader	After Update	AFTER UPDATE trigger that updates the RevisionNumber and ModifiedDate columns in the SalesOrderHeader table. Updates the SalesYTD column in the SalesPerson and SalesTerritory tables.

```

CREATE TRIGGER [Sales].[uSalesOrderHeader] ON [Sales].[SalesOrderHeader]
AFTER UPDATE NOT FOR REPLICATION AS
BEGIN
    DECLARE @Count int;
    SET @Count = @@ROWCOUNT;
    IF @Count = 0
        RETURN;
    SET NOCOUNT ON;
    BEGIN TRY
        -- Update RevisionNumber for modification of any field EXCEPT the Status.
        IF NOT UPDATE([Status])
        BEGIN
            UPDATE [Sales].[SalesOrderHeader]
            SET [Sales].[SalesOrderHeader].[RevisionNumber] =
                [Sales].[SalesOrderHeader].[RevisionNumber] + 1
            WHERE [Sales].[SalesOrderHeader].[SalesOrderID] IN
                (SELECT inserted.[SalesOrderID] FROM inserted);
        END;
        -- Update the SalesPerson SalesYTD when SubTotal is updated
        IF UPDATE([SubTotal])
        BEGIN
            DECLARE @StartDate datetime,
                    @EndDate datetime;
            SET @StartDate = [dbo].[ufnGetAccountingStartDate]();
            SET @EndDate = [dbo].[ufnGetAccountingEndDate]();
            UPDATE [Sales].[SalesPerson]
            SET [Sales].[SalesPerson].[SalesYTD] =
                (SELECT SUM([Sales].[SalesOrderHeader].[SubTotal])
                 FROM [Sales].[SalesOrderHeader]
                 WHERE [Sales].[SalesPerson].[BusinessEntityID] = [Sales].[SalesOrderHeader].[SalesPersonID]
                   AND ([Sales].[SalesOrderHeader].[Status] = 5) -- Shipped
                   AND [Sales].[SalesOrderHeader].[OrderDate] BETWEEN @StartDate AND @EndDate)
            WHERE [Sales].[SalesPerson].[BusinessEntityID]
                IN (SELECT DISTINCT inserted.[SalesPersonID] FROM inserted
                     WHERE inserted.[OrderDate] BETWEEN @StartDate AND @EndDate);
        END;
        -- Update the SalesTerritory SalesYTD when SubTotal is updated
        UPDATE [Sales].[SalesTerritory]
        SET [Sales].[SalesTerritory].[SalesYTD] =
            (SELECT SUM([Sales].[SalesOrderHeader].[SubTotal])
             FROM [Sales].[SalesOrderHeader]
             WHERE [Sales].[SalesTerritory].[TerritoryID] = [Sales].[SalesOrderHeader].[TerritoryID]
               AND ([Sales].[SalesOrderHeader].[Status] = 5) -- Shipped
               AND [Sales].[SalesOrderHeader].[OrderDate] BETWEEN @StartDate AND @EndDate)
        WHERE [Sales].[SalesTerritory].[TerritoryID]
            IN (SELECT DISTINCT inserted.[TerritoryID] FROM inserted
                 WHERE inserted.[OrderDate] BETWEEN @StartDate AND @EndDate);
    END;
    END TRY
    BEGIN CATCH
        EXECUTE [dbo].[uspPrintError];
        -- Rollback any active or uncommittable transactions before
        -- inserting information in the ErrorLog
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END
        EXECUTE [dbo].[uspLogError];
    END CATCH;
END;

```

## Uses

Name
<b>Sales.SalesOrderHeader</b>
↳ Sales.SalesOrderHeader
→ Person.Address
→ Person.Address
→ Purchasing.ShipMethod
→ Sales.CreditCard
→ Sales.CurrencyRate
→ Sales.Customer
→ Sales.SalesPerson
→ Sales.SalesTerritory
<b>Sales.uSalesOrderHeader</b>
↳ Sales.SalesOrderHeader
↳ Sales.SalesPerson
↳ Sales.SalesTerritory
↳ dbo.uspLogError
↳ dbo.uspPrintError
↳ dbo.ufnGetAccountingEndDate
↳ dbo.ufnGetAccountingStartDate

## Used By

Name
<b>Sales.SalesOrderHeader</b>
↳ Sales.SalesOrderHeader
↳ Sales.vSalesPersonSalesByFiscalYears
↳ Sales.iduSalesOrderDetail
↳ Sales.uSalesOrderHeader
→ Sales.SalesOrderDetail
→ Sales.SalesOrderHeaderSalesReason

## 8.8.1.9. Table: Sales.SalesOrderHeaderSalesReason

**Sample field: Status:** Active

Cross-reference table mapping sales orders to sales reason codes.

### Columns

		Name	Data type	Description / Attributes
█	🔑	SalesOrderID	int	Primary key. Foreign key to SalesOrderHeader.SalesOrderID. <b>References:</b> Sales.SalesOrderHeader
█	🔑	SalesReasonID	int	Primary key. Foreign key to SalesReason.SalesReasonID. <b>References:</b> Sales.SalesReason
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

	Table	Join	Title / Name / Description
→	Sales.SalesOrderHeader	Sales.SalesOrderHeaderSalesReason SalesOrderID = Sales.SalesOrderHeaderSalesOrderID	FK_SalesOrderHeaderSalesReason_SalesOrderHeader_SalesOrderID Foreign key constraint referencing SalesOrderHeader.SalesOrderID.
→	Sales.SalesReason	Sales.SalesOrderHeaderSalesReason SalesReasonID = Sales.SalesReasonSalesReasonID	FK_SalesOrderHeaderSalesReason_SalesReason_SalesReasonID Foreign key constraint referencing SalesReason.SalesReasonID.

### Unique keys

	Columns	Name / Description
🔑	SalesOrderID, SalesReasonID	PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID Primary key (clustered) constraint

### Uses

	Name
█	Sales.SalesOrderHeaderSalesReason
→	Sales.SalesOrderHeader
→	Sales.SalesReason

## 8.8.1.10. Table: Sales.SalesPerson

**Sample field: Status:** Active

Sales representative current information.

Columns

		Name	Data type	Description / Attributes
	PK	BusinessEntityID	int	Primary key for SalesPerson records. Foreign key to Employee.BusinessEntityID <b>References:</b> HumanResources.Employee
		TerritoryID	int	Territory currently assigned to. Foreign key to SalesTerritory.SalesTerritoryID. <b>Nullable</b> <b>References:</b> Sales.SalesTerritory
		SalesQuota	money	Projected yearly sales. <b>Nullable</b>
		Bonus	money	Bonus due if quota is met. <b>Default:</b> 0.00
		CommissionPct	smallmoney	Commision percent received per sale. <b>Default:</b> 0.00
		SalesYTD	money	Sales total year to date. <b>Default:</b> 0.00
		SalesLastYear	money	Sales total of previous year. <b>Default:</b> 0.00
	PK	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

Links to

Table		Join	Title / Name / Description
→	HumanResources.Employee	<b>Sales.SalesPerson</b> BusinessEntityID = HumanResources.EmployeeBusinessEntityID	FK_SalesPerson_Employee_BusinessEntityID Foreign key constraint referencing Employee.EmployeeID.
→	Sales.SalesTerritory	<b>Sales.SalesPerson</b> TerritoryID = Sales.SalesTerritoryTerritoryID	FK_SalesPerson_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.

Linked from

Table		Join	Title / Name / Description
→	Sales.SalesOrderHeader	<b>Sales.SalesPerson</b> BusinessEntityID = Sales.SalesOrderHeaderSalesPersonID	FK_SalesOrderHeader_SalesPerson_SalesPersonID Foreign key constraint referencing SalesPerson.SalesPersonID.
→	Sales.SalesPersonQuotaHistory	<b>Sales.SalesPerson</b> BusinessEntityID = Sales.SalesPersonQuotaHistoryBusinessEntityID	FK_SalesPersonQuotaHistory_SalesPerson_BusinessEntityID Foreign key constraint referencing SalesPerson.SalesPersonID.
→	Sales.SalesTerritoryHistory	<b>Sales.SalesPerson</b> BusinessEntityID = Sales.SalesTerritoryHistoryBusinessEntityID	FK_SalesTerritoryHistory_SalesPerson_BusinessEntityID Foreign key constraint referencing SalesPerson.SalesPersonID.
→	Sales.Store	<b>Sales.SalesPerson</b> BusinessEntityID = Sales.StoreSalesPersonID	FK_Store_SalesPerson_SalesPersonID Foreign key constraint referencing SalesPerson.SalesPersonID

## Unique keys

Columns		Name / Description
	BusinessEntityID	PK_SalesPerson_BusinessEntityID Primary key (clustered) constraint
	rowguid	AK_SalesPerson_rowguid Unique nonclustered index. Used to support replication samples.

## Uses

Name	
	Sales.SalesPerson
→	HumanResources.Employee
→	Sales.SalesTerritory

## Used By

Name	
	Sales.SalesPerson
↑	Sales.vSalesPerson
↑	Sales.vSalesPersonSalesByFiscalYears
⚡	Sales.uSalesOrderHeader
→	Sales.SalesOrderHeader
→	Sales.SalesPersonQuotaHistory
→	Sales.SalesTerritoryHistory
→	Sales.Store

## 8.8.1.11. Table: Sales.SalesPersonQuotaHistory

**Sample field: Status:** Active

Sales performance tracking.

### Columns

		Name	Data type	Description / Attributes
█	🔑	BusinessEntityID	int	Sales person identification number. Foreign key to SalesPerson.BusinessEntityID. <b>References:</b> Sales.SalesPerson
█	🔑	QuotaDate	datetime	Sales quota date.
█		SalesQuota	money	Sales quota amount.
█	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Sales.SalesPerson	Sales.SalesPersonQuotaHistoryBusiness sEntityID = Sales.SalesPersonBusinessEntityID	FK_SalesPersonQuotaHistory_SalesPerson_BusinessEntityID Foreign key constraint referencing SalesPerson.SalesPersonID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID, QuotaDate	PK_SalesPersonQuotaHistory_BusinessEntityID_QquotaDate Primary key (clustered) constraint
🔑	rowguid	AK_SalesPersonQuotaHistory_rowguid Unique nonclustered index. Used to support replication samples.

### Uses

Name	
█	Sales.SalesPersonQuotaHistory
→	Sales.SalesPerson

## 8.8.1.12. Table: Sales.SalesReason

**Sample field: Status:** Active

Lookup table of customer purchase reasons.

### Columns

Name		Data type	Description / Attributes
	SalesReasonID	int	Primary key for SalesReason records. <b>Identity / Auto increment</b>
	Name	Name: nvarchar(50)	Sales reason description.
	ReasonType	Name: nvarchar(50)	Category the sales reason belongs to.
	ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
	Sales.SalesOrderHeaderSalesReason	<b>Sales.SalesReason</b> SalesReasonID = Sales.SalesOrderHeaderSalesReasonSalesReasonID	FK_SalesOrderHeaderSalesReason_SalesReason_SalesReasonID Foreign key constraint referencing SalesReason.SalesReasonID.

### Unique keys

Columns		Name / Description
	SalesReasonID	<b>PK_SalesReason_SalesReasonID</b> Primary key (clustered) constraint

### Used By

Name	
	Sales.SalesReason
	Sales.SalesOrderHeaderSalesReason

### 8.8.1.13. Table: Sales.SalesTaxRate

**Sample field: Status:** Active

Tax rate lookup table.

#### Columns

		Name	Data type	Description / Attributes
■	🔑	SalesTaxRateID	int	Primary key for SalesTaxRate records. <b>Identity / Auto increment</b>
■	🔑	StateProvinceID	int	State, province, or country/region the sales tax applies to. <b>References:</b> Person.StateProvince
■	🔑	TaxType	tinyint	1 = Tax applied to retail transactions, 2 = Tax applied to wholesale transactions, 3 = Tax applied to all sales (retail and wholesale) transactions.
■		TaxRate	smallmoney	Tax rate amount. <b>Default:</b> 0.00
■		Name	Name: nvarchar(50)	Tax rate description.
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

#### Links to

Table		Join	Title / Name / Description
→	Person.StateProvince	Sales.SalesTaxRate StateProvinceID = Person.StateProvinceStateProvinceID	FK_SalesTaxRate_StateProvince_StateProvinceID Foreign key constraint referencing StateProvince.StateProvinceID.

#### Unique keys

Columns		Name / Description
🔑	SalesTaxRateID	PK_SalesTaxRate_SalesTaxRateID Primary key (clustered) constraint
🔑	rowguid	AK_SalesTaxRate_rowguid Unique nonclustered index. Used to support replication samples.
🔑	StateProvinceID, TaxType	AK_SalesTaxRate_StateProvinceID_TaxType Unique nonclustered index.

#### Uses

Name	
grid	Sales.SalesTaxRate
→	Person.StateProvince

## 8.8.1.14. Table: Sales.SalesTerritory

**Sample field: Status:** Active

Sales territory lookup table.

### Columns

		Name	Data type	Description / Attributes
■	🔑	TerritoryID	int	Primary key for SalesTerritory records. <b>Identity / Auto increment</b>
■	🔑	Name	Name: nvarchar(50)	Sales territory description
■		CountryRegionCode	nvarchar(3)	ISO standard country or region code. Foreign key to CountryRegion.CountryRegionCode. <b>References:</b> Person.CountryRegion
■		Group	nvarchar(50)	Geographic area to which the sales territory belong.
■		SalesYTD	money	Sales in the territory year to date. <b>Default:</b> 0.00
■		SalesLastYear	money	Sales in the territory the previous year. <b>Default:</b> 0.00
■		CostYTD	money	Business costs in the territory year to date. <b>Default:</b> 0.00
■		CostLastYear	money	Business costs in the territory the previous year. <b>Default:</b> 0.00
■	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
■		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.CountryRegion	Sales.SalesTerritoryCountryRegionCode = Person.CountryRegionCountryRegionCode	FK_SalesTerritory_CountryRegion_CountryRegionCode Foreign key constraint referencing CountryRegion.CountryRegionCode.

### Linked from

Table		Join	Title / Name / Description
←	Sales.Customer	Sales.SalesTerritoryTerritoryID = Sales.CustomerTerritoryID	FK_Customer_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.
←	Sales.SalesOrderHeader	Sales.SalesTerritoryTerritoryID = Sales.SalesOrderHeaderTerritoryID	FK_SalesOrderHeader_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.
←	Sales.SalesPerson	Sales.SalesTerritoryTerritoryID = Sales.SalesPersonTerritoryID	FK_SalesPerson_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.
←	Sales.SalesTerritoryHistory	Sales.SalesTerritoryTerritoryID = Sales.SalesTerritoryHistoryTerritoryID	FK_SalesTerritoryHistory_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.
←	Person.StateProvince	Sales.SalesTerritoryTerritoryID = Person.StateProvinceTerritoryID	FK_StateProvince_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.

## Unique keys

Columns		Name / Description
	TerritoryID	PK_SalesTerritory_TerritoryID Primary key (clustered) constraint
	Name	AK_SalesTerritory_Name Unique nonclustered index.
	rowguid	AK_SalesTerritory_rowguid Unique nonclustered index. Used to support replication samples.

## Uses

Name
Sales.SalesTerritory
→ Person.CountryRegion

## Used By

Name
Sales.SalesTerritory
↳ Sales.vSalesPerson
↳ Sales.vSalesPersonSalesByFiscalYears
↳ Sales.uSalesOrderHeader
← Person.StateProvince
← Sales.Customer
← Sales.SalesOrderHeader
← Sales.SalesPerson
← Sales.SalesTerritoryHistory

## 8.8.1.15. Table: Sales.SalesTerritoryHistory

**Sample field: Status:** Active

Sales representative transfers to other sales territories.

### Columns

		Name	Data type	Description / Attributes
		BusinessEntityID	int	Primary key. The sales rep. Foreign key to SalesPerson.BusinessEntityID. <b>References:</b> Sales.SalesPerson
		TerritoryID	int	Primary key. Territory identification number. Foreign key to SalesTerritory.SalesTerritoryID. <b>References:</b> Sales.SalesTerritory
		StartDate	datetime	Primary key. Date the sales representative started work in the territory.
		EndDate	datetime	Date the sales representative left work in the territory. <b>Nullable</b>
		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Sales.SalesPerson	<b>Sales.SalesTerritoryHistory</b> BusinessEntityID = Sales.SalesPersonBusinessEntityID	FK_SalesTerritoryHistory_SalesPerson_BusinessEntityID Foreign key constraint referencing SalesPerson.SalesPersonID.
→	Sales.SalesTerritory	<b>Sales.SalesTerritoryHistory</b> TerritoryID = Sales.SalesTerritoryTerritoryID	FK_SalesTerritoryHistory_SalesTerritory_TerritoryID Foreign key constraint referencing SalesTerritory.TerritoryID.

### Unique keys

Columns		Name / Description
	BusinessEntityID, StartDate, TerritoryID	<b>PK_SalesTerritoryHistory_BusinessEntityID_StartDate_TerritoryID</b> Primary key (clustered) constraint
	rowguid	<b>AK_SalesTerritoryHistory_rowguid</b> Unique nonclustered index. Used to support replication samples.

### Uses

Name	
	Sales.SalesTerritoryHistory
→	Sales.SalesPerson
→	Sales.SalesTerritory

## 8.8.1.16. Table: Sales.ShoppingCartItem

**Sample field: Status:** Active

Contains online customer orders until the order is submitted or cancelled.

### Columns

		Name	Data type	Description / Attributes
		ShoppingCartItemID	int	Primary key for ShoppingCartItem records. <b>Identity / Auto increment</b>
		ShoppingCartID	nvarchar(50)	Shopping cart identification number.
		Quantity	int	Product quantity ordered. <b>Default:</b> 1
		ProductID	int	Product ordered. Foreign key to Product.ProductID. <b>References:</b> Production.Product
		DateCreated	datetime	Date the time the record was created. <b>Default:</b> getdate()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
	Production.Product	Sales.ShoppingCartItem.ProductID = Production.Product.ProductID	FK_ShoppingCartItem_Product_ProductID Foreign key constraint referencing Product.ProductID.

### Unique keys

Columns		Name / Description
	ShoppingCartItemID	PK_ShoppingCartItem_ShoppingCartItemID Primary key (clustered) constraint

### Uses

Name	
	Sales.ShoppingCartItem
	Production.Product

## 8.8.1.17. Table: Sales.SpecialOffer

**Sample field: Status:** Active

Sale discounts lookup table.

### Columns

		Name	Data type	Description / Attributes
█	🔑	SpecialOfferID		Primary key for SpecialOffer records. <b>Identity / Auto increment</b>
█		Description		Discount description.
█		DiscountPct		Discount percentage. <b>Default:</b> 0.00
█		Type		Discount type category.
█		Category		Group the discount applies to such as Reseller or Customer.
█		StartDate		Discount start date.
█		EndDate		Discount end date.
█		MinQty		Minimum discount percent allowed. <b>Default:</b> 0
█		MaxQty		Maximum discount percent allowed. <b>Nullable</b>
█	🔑	rowguid		ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
█		ModifiedDate		Date and time the record was last updated. <b>Default:</b> getdate()

### Linked from

Table		Join	Title / Name / Description
→	Sales.SpecialOfferProduct	Sales.SpecialOfferSpecialOfferID = Sales.SpecialOfferProductSpecialOfferID	FK_SpecialOfferProduct_SpecialOffer_SpecialOfferID Foreign key constraint referencing SpecialOffer.SpecialOfferID.

### Unique keys

Columns		Name / Description
🔑	SpecialOfferID	PK_SpecialOffer_SpecialOfferID Primary key (clustered) constraint
🔑	rowguid	AK_SpecialOffer_rowguid Unique nonclustered index. Used to support replication samples.

### Used By

Name	
█	Sales.SpecialOffer
→	Sales.SpecialOfferProduct

## 8.8.1.18. Table: Sales.SpecialOfferProduct

**Sample field: Status:** Active

Cross-reference table mapping products to special offer discounts.

### Columns

		Name	Data type	Description / Attributes
		SpecialOfferID	int	Primary key for SpecialOfferProduct records. <b>References:</b> Sales.SpecialOffer
		ProductID	int	Product identification number. Foreign key to Product.ProductID. <b>References:</b> Production.Product
		rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
	Production.Product	Sales.SpecialOfferProductProductID = Production.ProductProductID	FK_SpecialOfferProduct_Product_ProductID Foreign key constraint referencing Product.ProductID.
	Sales.SpecialOffer	Sales.SpecialOfferProductSpecialOfferID = Sales.SpecialOfferSpecialOfferID	FK_SpecialOfferProduct_SpecialOffer_SpecialOfferID Foreign key constraint referencing SpecialOffer.SpecialOfferID.

### Linked from

Table		Join	Title / Name / Description
	Sales.SalesOrderDetail	Sales.SpecialOfferProductSpecialOfferID = Sales.SalesOrderDetailSpecialOfferID, Sales.SpecialOfferProductProductID = Sales.SalesOrderDetailProductID	FK_SalesOrderDetail_SpecialOfferProduct_SpecialOfferIDProductID Foreign key constraint referencing SpecialOfferProduct.SpecialOfferIDProductID.

### Unique keys

Columns		Name / Description
	SpecialOfferID, ProductID	PK_SpecialOfferProduct_SpecialOfferID_ProductID Primary key (clustered) constraint
	rowguid	AK_SpecialOfferProduct_rowguid Unique nonclustered index. Used to support replication samples.

### Uses

		Name
	Sales.SpecialOfferProduct	
	Production.Product	
	Sales.SpecialOffer	

### Used By

		Name
	Sales.SpecialOfferProduct	
	Sales.SalesOrderDetail	

## 8.8.1.19. Table: Sales.Store

**Sample field: Status:** Active

Customers (resellers) of Adventure Works products.

### Columns

		Name	Data type	Description / Attributes
█	🔑	BusinessEntityID	int	Primary key. Foreign key to Customer.BusinessEntityID. <b>References:</b> Person.BusinessEntity
█		Name	Name: nvarchar(50)	Name of the store.
█		SalesPersonID	int	ID of the sales person assigned to the customer. Foreign key to SalesPerson.BusinessEntityID. <b>Nullable</b> <b>References:</b> Sales.SalesPerson
█		Demographics	xml	Demographic information about the store such as the number of employees, annual sales and store type. <b>Nullable</b>
█	🔑	rowguid	uniqueidentifier	ROWGUIDCOL number uniquely identifying the record. Used to support a merge replication sample. <b>Default:</b> newid()
█		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

### Links to

Table		Join	Title / Name / Description
→	Person.BusinessEntity	<b>Sales.Store</b> BusinessEntityID = Person.BusinessEntityBusinessEntityID	FK_Store_BusinessEntity_BusinessEntityID Foreign key constraint referencing BusinessEntity.BusinessEntityID
→	Sales.SalesPerson	<b>Sales.Store</b> SalesPersonID = Sales.SalesPersonBusinessEntityID	FK_Store_SalesPerson_SalesPersonID Foreign key constraint referencing SalesPerson.SalesPersonID

### Linked from

Table		Join	Title / Name / Description
←	Sales.Customer	<b>Sales.Store</b> BusinessEntityID = Sales.CustomerStoreID	FK_Customer_Store_StoreID Foreign key constraint referencing Store.BusinessEntityID.

### Unique keys

Columns		Name / Description
🔑	BusinessEntityID	PK_Store_BusinessEntityID Primary key (clustered) constraint
🔑	rowguid	AK_Store_rowguid Unique nonclustered index. Used to support replication samples.

### Uses

Name	
█	<b>Sales.Store</b>
→	Person.BusinessEntity
→	Sales.SalesPerson

## Used By

Name
Sales.Store
Sales.vStoreWithAddresses
Sales.vStoreWithContacts
Sales.vStoreWithDemographics
dbo.ufnGetContactInformation
Sales.Customer

TRIAL

## 8.8.2. Views

### 8.8.2.1. View: Sales.vIndividualCustomer

**Sample field: Status:** Active

Individual customers (names and addresses) that purchase Adventure Works Cycles products online.

#### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Title	nvarchar(8)	<b>Nullable</b>
3	FirstName	Name: nvarchar(50)	
4	MiddleName	Name: nvarchar(50)	<b>Nullable</b>
5	LastName	Name: nvarchar(50)	
6	Suffix	nvarchar(10)	<b>Nullable</b>
7	PhoneNumber	Phone: nvarchar(25)	<b>Nullable</b>
8	PhoneNumberType	Name: nvarchar(50)	<b>Nullable</b>
9	EmailAddress	nvarchar(50)	<b>Nullable</b>
10	EmailPromotion	int	
11	AddressType	Name: nvarchar(50)	
12	AddressLine1	nvarchar(60)	
13	AddressLine2	nvarchar(60)	<b>Nullable</b>
14	City	nvarchar(30)	
15	StateProvinceName	Name: nvarchar(50)	
16	PostalCode	nvarchar(15)	
17	CountryRegionName	Name: nvarchar(50)	
18	Demographics	xml	<b>Nullable</b>

#### Uses

	Name
1	Sales.vIndividualCustomer
2	Person.Address
3	Person.AddressType
4	Person.BusinessEntityAddress
5	Person.CountryRegion
6	Person.EmailAddress
7	Person.Person
8	Person.PersonPhone
9	Person.PhoneNumberType
10	Person.StateProvince
11	Sales.Customer

## Script

```
CREATE VIEW [Sales].[vIndividualCustomer]
AS
SELECT
    p.[BusinessEntityID]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,pp.[PhoneNumber]
    ,pnt.[Name] AS [PhoneNumberType]
    ,ea.[EmailAddress]
    ,p.[EmailPromotion]
    ,at.[Name] AS [AddressType]
    ,a.[AddressLine1]
    ,a.[AddressLine2]
    ,a.[City]
    ,[StateProvinceName] = sp.[Name]
    ,a.[PostalCode]
    ,[CountryRegionName] = cr.[Name]
    ,p.[Demographics]
FROM [Person].[Person] p
INNER JOIN [Person].[BusinessEntityAddress] bea
ON bea.[BusinessEntityID] = p.[BusinessEntityID]
INNER JOIN [Person].[Address] a
ON a.[AddressID] = bea.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
INNER JOIN [Person].[AddressType] at
ON at.[AddressTypeID] = bea.[AddressTypeID]
    INNER JOIN [Sales].[Customer] c
    ON c.[PersonID] = p.[BusinessEntityID]
    LEFT OUTER JOIN [Person].[EmailAddress] ea
    ON ea.[BusinessEntityID] = p.[BusinessEntityID]
    LEFT OUTER JOIN [Person].[PersonPhone] pp
    ON pp.[BusinessEntityID] = p.[BusinessEntityID]
    LEFT OUTER JOIN [Person].[PhoneNumberType] pnt
    ON pnt.[PhoneNumberTypeID] = pp.[PhoneNumberTypeID]
WHERE c.StoreID IS NULL;
```

TRIAL

## 8.8.2.2. View: Sales.vPersonDemographics

### Sample field: Status: Active

Displays the content from each element in the xml column Demographics for each customer in the Person.Person table.

### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	TotalPurchaseYTD	money	Nullable
3	DateFirstPurchase	datetime	Nullable
4	BirthDate	datetime	Nullable
5	MaritalStatus	nvarchar(1)	Nullable
6	YearlyIncome	nvarchar(30)	Nullable
7	Gender	nvarchar(1)	Nullable
8	TotalChildren	int	Nullable
9	NumberChildrenAtHome	int	Nullable
10	Education	nvarchar(30)	Nullable
11	Occupation	nvarchar(30)	Nullable
12	HomeOwnerFlag	bit	Nullable
13	NumberCarsOwned	int	Nullable

### Uses

	Name
1	Sales.vPersonDemographics
2	Person.Person
3	IndividualSurvey.ref.value

## Script

```
CREATE VIEW [Sales].[vPersonDemographics]
AS
SELECT
    p.[BusinessEntityID]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    TotalPurchaseYTD[1]', 'money') AS [TotalPurchaseYTD]
    ,CONVERT(datetime, REPLACE([IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    DateFirstPurchase[1]', 'nvarchar(20)'), 'Z', ''), 101) AS [DateFirstPurchase]
    ,CONVERT(datetime, REPLACE([IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    BirthDate[1]', 'nvarchar(20)'), 'Z', ''), 101) AS [BirthDate]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    MaritalStatus[1]', 'nvarchar(1)') AS [MaritalStatus]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    YearlyIncome[1]', 'nvarchar(30)') AS [YearlyIncome]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    Gender[1]', 'nvarchar(1)') AS [Gender]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    TotalChildren[1]', 'integer') AS [TotalChildren]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    NumberChildrenAtHome[1]', 'integer') AS [NumberChildrenAtHome]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    Education[1]', 'nvarchar(30)') AS [Education]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    Occupation[1]', 'nvarchar(30)') AS [Occupation]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    HomeOwnerFlag[1]', 'bit') AS [HomeOwnerFlag]
    ,[IndividualSurvey].[ref].[value](N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
    NumberCarsOwned[1]', 'integer') AS [NumberCarsOwned]
FROM [Person].[Person] p
CROSS APPLY p.[Demographics].nodes(N'declare default element namespace
"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/IndividualSurvey";
/IndividualSurvey') AS [IndividualSurvey](ref)
WHERE [Demographics] IS NOT NULL;
```

### 8.8.2.3. View: Sales.vSalesPerson

**Sample field: Status:** Active

Sales representatives (names and addresses) and their sales-related information.

#### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Title	nvarchar(8)	<b>Nullable</b>
3	FirstName	Name: nvarchar(50)	
4	MiddleName	Name: nvarchar(50)	<b>Nullable</b>
5	LastName	Name: nvarchar(50)	
6	Suffix	nvarchar(10)	<b>Nullable</b>
7	JobTitle	nvarchar(50)	
8	PhoneNumber	Phone: nvarchar(25)	<b>Nullable</b>
9	PhoneNumberType	Name: nvarchar(50)	<b>Nullable</b>
10	EmailAddress	nvarchar(50)	<b>Nullable</b>
11	EmailPromotion	int	
12	AddressLine1	nvarchar(60)	
13	AddressLine2	nvarchar(60)	<b>Nullable</b>
14	City	nvarchar(30)	
15	StateProvinceName	Name: nvarchar(50)	
16	PostalCode	nvarchar(15)	
17	CountryRegionName	Name: nvarchar(50)	
18	TerritoryName	Name: nvarchar(50)	<b>Nullable</b>
19	TerritoryGroup	nvarchar(50)	<b>Nullable</b>
20	SalesQuota	money	<b>Nullable</b>
21	SalesYTD	money	
22	SalesLastYear	money	

#### Uses

	Name
1	Sales.vSalesPerson
2	HumanResources.Employee
3	Person.Address
4	Person.BusinessEntityAddress
5	Person.CountryRegion
6	Person.EmailAddress
7	Person.Person
8	Person.PersonPhone
9	Person.PhoneNumberType

## Name

Person.StateProvince
Sales.SalesPerson
Sales.SalesTerritory

## Script

```
CREATE VIEW [Sales].[vSalesPerson]
AS
SELECT
    s.[BusinessEntityID]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,e.[JobTitle]
    ,pp.[PhoneNumber]
    ,pnt.[Name] AS [PhoneNumberType]
    ,ea.[EmailAddress]
    ,p.[EmailPromotion]
    ,a.[AddressLine1]
    ,a.[AddressLine2]
    ,a.[City]
    ,[StateProvinceName] = sp.[Name]
    ,a.[PostalCode]
    ,[CountryRegionName] = cr.[Name]
    ,[TerritoryName] = st.[Name]
    ,[TerritoryGroup] = st.[Group]
    ,s.[SalesQuota]
    ,s.[SalesYTD]
    ,s.[SalesLastYear]
FROM [Sales].[SalesPerson] s
INNER JOIN [HumanResources].[Employee] e
ON e.[BusinessEntityID] = s.[BusinessEntityID]
    INNER JOIN [Person].[Person] p
        ON p.[BusinessEntityID] = s.[BusinessEntityID]
INNER JOIN [Person].[BusinessEntityAddress] bea
ON bea.[BusinessEntityID] = s.[BusinessEntityID]
INNER JOIN [Person].[Address] a
ON a.[AddressID] = bea.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
LEFT OUTER JOIN [Sales].[SalesTerritory] st
ON st.[TerritoryID] = s.[TerritoryID]
    LEFT OUTER JOIN [Person].[EmailAddress] ea
        ON ea.[BusinessEntityID] = p.[BusinessEntityID]
    LEFT OUTER JOIN [Person].[PersonPhone] pp
        ON pp.[BusinessEntityID] = p.[BusinessEntityID]
    LEFT OUTER JOIN [Person].[PhoneNumberType] pnt
        ON pnt.[PhoneNumberTypeID] = pp.[PhoneNumberTypeID];
```

## 8.8.2.4. View: Sales.vSalesPersonSalesByFiscalYears

**Sample field: Status:** Active

Uses PIVOT to return aggregated sales information for each sales representative.

### Columns

	Name	Data type	Description / Attributes
1	SalesPersonID	int	Nullable
2	FullName	nvarchar(152)	Nullable
3	JobTitle	nvarchar(50)	
4	SalesTerritory	Name: nvarchar(50)	
5	2002	money	Nullable
6	2003	money	Nullable
7	2004	money	Nullable

### Uses

Name
Sales.vSalesPersonSalesByFiscalYears
HumanResources.Employee
Person.Person
Sales.SalesOrderHeader
Sales.SalesPerson
Sales.SalesTerritory

### Script

```

CREATE VIEW [Sales].[vSalesPersonSalesByFiscalYears]
AS
SELECT
    pvt.[SalesPersonID]
    ,pvt.[FullName]
    ,pvt.[JobTitle]
    ,pvt.[SalesTerritory]
    ,pvt.[2002]
    ,pvt.[2003]
    ,pvt.[2004]
FROM (SELECT
        soh.[SalesPersonID]
        ,p.[FirstName] + ' ' + COALESCE(p.[MiddleName], '') + ' ' + p.[LastName] AS [FullName]
        ,e.[JobTitle]
        ,st.[Name] AS [SalesTerritory]
        ,soh.[SubTotal]
        ,YEAR(DATEADD(m, 6, soh.[OrderDate])) AS [FiscalYear]
    FROM [Sales].[SalesPerson] sp
    INNER JOIN [Sales].[SalesOrderHeader] soh
    ON sp.[BusinessEntityID] = soh.[SalesPersonID]
    INNER JOIN [Sales].[SalesTerritory] st
    ON sp.[TerritoryID] = st.[TerritoryID]
    INNER JOIN [HumanResources].[Employee] e
    ON soh.[SalesPersonID] = e.[BusinessEntityID]
        INNER JOIN [Person].[Person] p
        ON p.[BusinessEntityID] = sp.[BusinessEntityID]
    ) AS soh
PIVOT
(
    SUM([SubTotal])
    FOR [FiscalYear]
    IN ([2002], [2003], [2004])
) AS pvt;

```

## 8.8.2.5. View: Sales.vStoreWithAddresses

### Sample field: Status: Active

Stores (including store addresses) that sell Adventure Works Cycles products to consumers.

### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Name	Name: nvarchar(50)	
3	AddressType	Name: nvarchar(50)	
4	AddressLine1	nvarchar(60)	
5	AddressLine2	nvarchar(60)	Nullable
6	City	nvarchar(30)	
7	StateProvinceName	Name: nvarchar(50)	
8	PostalCode	nvarchar(15)	
9	CountryRegionName	Name: nvarchar(50)	

### Uses

Name	
1	Sales.vStoreWithAddresses
2	Person.Address
3	Person.AddressType
4	Person.BusinessEntityAddress
5	Person.CountryRegion
6	Person.StateProvince
7	Sales.Store

### Script

```

CREATE VIEW [Sales].[vStoreWithAddresses] AS
SELECT
    s.[BusinessEntityID]
    ,s.[Name]
    ,at.[Name] AS [AddressType]
    ,a.[AddressLine1]
    ,a.[AddressLine2]
    ,a.[City]
    ,sp.[Name] AS [StateProvinceName]
    ,a.[PostalCode]
    ,cr.[Name] AS [CountryRegionName]
FROM [Sales].[Store] s
INNER JOIN [Person].[BusinessEntityAddress] bea
ON bea.[BusinessEntityID] = s.[BusinessEntityID]
INNER JOIN [Person].[Address] a
ON a.[AddressID] = bea.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
INNER JOIN [Person].[AddressType] at
ON at.[AddressTypeID] = bea.[AddressTypeID];

```

## 8.8.2.6. View: Sales.vStoreWithContacts

**Sample field: Status:** Active

Stores (including store contacts) that sell Adventure Works Cycles products to consumers.

### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Name	Name: nvarchar(50)	
3	ContactType	Name: nvarchar(50)	
4	Title	nvarchar(8)	<b>Nullable</b>
5	FirstName	Name: nvarchar(50)	
6	MiddleName	Name: nvarchar(50)	<b>Nullable</b>
7	LastName	Name: nvarchar(50)	
8	Suffix	nvarchar(10)	<b>Nullable</b>
9	PhoneNumber	Phone: nvarchar(25)	<b>Nullable</b>
10	PhoneNumberType	Name: nvarchar(50)	<b>Nullable</b>
11	EmailAddress	nvarchar(50)	<b>Nullable</b>
12	EmailPromotion	int	

### Uses

	Name
1	Sales.vStoreWithContacts
2	Person.BusinessEntityContact
3	Person.ContactType
4	Person.EmailAddress
5	Person.Person
6	Person.PersonPhone
7	Person.PhoneNumberType
8	Sales.Store

## Script

```
CREATE VIEW [Sales].[vStoreWithContacts] AS
SELECT
    s.[BusinessEntityID]
    ,s.[Name]
    ,ct.[Name] AS [ContactType]
    ,p.[Title]
    ,p.[FirstName]
    ,p.[MiddleName]
    ,p.[LastName]
    ,p.[Suffix]
    ,pp.[PhoneNumber]
        ,pnt.[Name] AS [PhoneNumberType]
    ,ea.[EmailAddress]
    ,p.[EmailPromotion]
FROM [Sales].[Store] s
INNER JOIN [Person].[BusinessEntityContact] bec
ON bec.[BusinessEntityID] = s.[BusinessEntityID]
    INNER JOIN [Person].[ContactType] ct
        ON ct.[ContactTypeID] = bec.[ContactTypeID]
    INNER JOIN [Person].[Person] p
        ON p.[BusinessEntityID] = bec.[PersonID]
LEFT OUTER JOIN [Person].[EmailAddress] ea
    ON ea.[BusinessEntityID] = p.[BusinessEntityID]
LEFT OUTER JOIN [Person].[PersonPhone] pp
    ON pp.[BusinessEntityID] = p.[BusinessEntityID]
LEFT OUTER JOIN [Person].[PhoneNumberType] pnt
    ON pnt.[PhoneNumberTypeID] = pp.[PhoneNumberTypeID];
```

TRIAL

## 8.8.2.7. View: Sales.vStoreWithDemographics

**Sample field: Status:** Active

Stores (including demographics) that sell Adventure Works Cycles products to consumers.

### Columns

	Name	Data type	Description / Attributes
1	BusinessEntityID	int	
2	Name	Name: nvarchar(50)	
3	AnnualSales	money	<b>Nullable</b>
4	AnnualRevenue	money	<b>Nullable</b>
5	BankName	nvarchar(50)	<b>Nullable</b>
6	BusinessType	nvarchar(5)	<b>Nullable</b>
7	YearOpened	int	<b>Nullable</b>
8	Specialty	nvarchar(50)	<b>Nullable</b>
9	SquareFeet	int	<b>Nullable</b>
10	Brands	nvarchar(30)	<b>Nullable</b>
11	Internet	nvarchar(30)	<b>Nullable</b>
12	NumberEmployees	int	<b>Nullable</b>

### Uses

	Name
1	Sales.vStoreWithDemographics
2	Sales.Store
3	s.Demographics.value

## Script

```
CREATE VIEW [Sales].[vStoreWithDemographics] AS
SELECT
    s.[BusinessEntityID]
    ,s.[Name]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey";
        (/StoreSurvey/AnnualSales)[1]', 'money') AS [AnnualSales]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/AnnualRevenue)[1]', 'money') AS [AnnualRevenue]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/BankName)[1]', 'nvarchar(50)') AS [BankName]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/BusinessType)[1]', 'nvarchar(5)') AS [BusinessType]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/YearOpened)[1]', 'integer') AS [YearOpened]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/Specialty)[1]', 'nvarchar(50)') AS [Specialty]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/SquareFeet)[1]', 'integer') AS [SquareFeet]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/Brands)[1]', 'nvarchar(30)') AS [Brands]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/Internet)[1]', 'nvarchar(30)') AS [Internet]
    ,s.[Demographics].value('declare default element namespace "http://schemas.microsoft.com/sqlserver/2004/07/adventure-
works/StoreSurvey';
        (/StoreSurvey/NumberEmployees)[1]', 'integer') AS [NumberEmployees]
FROM [Sales].[Store] s;
```

DATA

## 8.8.3. Functions

### 8.8.3.1. Function: dbo.ufnGetAccountingEndDate

Scalar function used in the uSalesOrderHeader trigger to set the starting account date.

#### Input/Output

	Name	Data type	Description
*@	Returns	datetime	

#### Used By

	Name
fx	dbo.ufnGetAccountingEndDate
trig	Sales.uSalesOrderHeader

#### Script

```
CREATE FUNCTION [dbo].[ufnGetAccountingEndDate] ()  
RETURNS [datetime]  
AS  
BEGIN  
    RETURN DATEADD(millisecond, -2, CONVERT(datetime, '20040701', 112));  
END;
```

### 8.8.3.2. Function: dbo.ufnGetAccountingStartDate

Scalar function used in the uSalesOrderHeader trigger to set the ending account date.

#### Input/Output

Name	Data type	Description
*@ Returns	datetime	

#### Used By

Name
dbo.ufnGetAccountingStartDate
Sales.uSalesOrderHeader

#### Script

```
CREATE FUNCTION [dbo].[ufnGetAccountingStartDate]()
RETURNS [datetime]
AS
BEGIN
    RETURN CONVERT(datetime, '20030701', 112);
END;
```

### 8.8.3.3. Function: dbo.ufnGetSalesOrderStatusText

Scalar function returning the text representation of the Status column in the SalesOrderHeader table.

#### Input/Output

	Name	Data type	Description
*@	Returns	nvarchar(15)	
*@	Status	tinyint	Input parameter for the scalar function ufnGetSalesOrderStatusText. Enter a valid integer.

#### Script

```
CREATE FUNCTION [dbo].[ufnGetSalesOrderStatusText] (@Status [tinyint])
RETURNS [nvarchar](15)
AS
-- Returns the sales order status text representation for the status value.
BEGIN
    DECLARE @ret [nvarchar](15);
    SET @ret =
        CASE @Status
            WHEN 1 THEN 'In process'
            WHEN 2 THEN 'Approved'
            WHEN 3 THEN 'Backordered'
            WHEN 4 THEN 'Rejected'
            WHEN 5 THEN 'Shipped'
            WHEN 6 THEN 'Cancelled'
            ELSE '** Invalid **'
        END;
    RETURN @ret
END;
```

#### 8.8.3.4. Function: dbo.ufnLeadingZeros

Scalar function used by the Sales.Customer table to help set the account number.

##### Input/Output

	Name	Data type	Description
*@	Returns	varchar(8)	
*@	Value	int	Input parameter for the scalar function ufnLeadingZeros. Enter a valid integer.

##### Used By

	Name
fx	dbo.ufnLeadingZeros
grid	Sales.Customer

##### Script

```
CREATE FUNCTION [dbo].[ufnLeadingZeros] (
    @Value int
)
RETURNS varchar(8)
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @ReturnValue varchar(8);
    SET @ReturnValue = CONVERT(varchar(8), @Value);
    SET @ReturnValue = REPLICATE('0', 8 - DATALENGTH(@ReturnValue)) + @ReturnValue;
    RETURN (@ReturnValue);
END;
```

TRIAL

## 8.9. Admin

### 8.9.1. Tables

#### 8.9.1.1. Table: dbo.AWBuildVersion

**Sample field: Status:** Active

Current version number of the AdventureWorks 2012 sample database.

##### Columns

		Name	Data type	Description / Attributes
		SystemInformationID	tinyint	Primary key for AWBuildVersion records. <b>Identity / Auto increment</b>
		Database Version	nvarchar(25)	Version number of the database in 9.yy.mm.dd.00 format.
		VersionDate	datetime	Date and time the record was last updated.
		ModifiedDate	datetime	Date and time the record was last updated. <b>Default:</b> getdate()

##### Unique keys

	Columns	Name / Description
	SystemInformationID	PK_AWBuildVersion_SystemInformationID Primary key (clustered) constraint

### 8.9.1.2. Table: dbo.DatabaseLog

**Sample field: Status:** Active

Audit table tracking all DDL changes made to the AdventureWorks database. Data is captured by the database trigger ddlDatabaseTriggerLog.

#### Columns

		Name	Data type	Description / Attributes
	DatabaseLogID	int	Primary key for DatabaseLog records. <b>Identity / Auto increment</b>	
	PostTime	datetime	The date and time the DDL change occurred.	
	DatabaseUser	nvarchar(128)	The user who implemented the DDL change.	
	Event	nvarchar(128)	The type of DDL statement that was executed.	
	Schema	nvarchar(128)	The schema to which the changed object belongs. <b>Nullable</b>	
	Object	nvarchar(128)	The object that was changed by the DDL statement. <b>Nullable</b>	
	TSQL	nvarchar(MAX)	The exact Transact-SQL statement that was executed.	
	XmlEvent	xml	The raw XML data generated by database trigger.	

#### Unique keys

Columns		Name / Description
	DatabaseLogID	PK_DatabaseLog_DatabaseLogID Primary key (nonclustered) constraint

### 8.9.1.3. Table: dbo.ErrorLog

**Sample field: Status:** Active

Audit table tracking errors in the AdventureWorks database that are caught by the CATCH block of a TRY...CATCH construct. Data is inserted by stored procedure dbo.uspLogError when it is executed from inside the CATCH block of a TRY...CATCH construct.

#### Columns

	Name	Data type	Description / Attributes
█	ErrorLogID	int	Primary key for ErrorLog records. <b>Identity / Auto increment</b>
█	ErrorTime	datetime	The date and time at which the error occurred. <b>Default:</b> getdate()
█	UserName	nvarchar(128)	The user who executed the batch in which the error occurred.
█	ErrorNumber	int	The error number of the error that occurred.
█	ErrorSeverity	int	The severity of the error that occurred. <b>Nullable</b>
█	ErrorState	int	The state number of the error that occurred. <b>Nullable</b>
█	ErrorProcedure	nvarchar(126)	The name of the stored procedure or trigger where the error occurred. <b>Nullable</b>
█	ErrorLine	int	The line number at which the error occurred. <b>Nullable</b>
█	ErrorMessage	nvarchar(4000)	The message text of the error that occurred.

#### Unique keys

	Columns	Name / Description
█	ErrorLogID	PK_ErrorLog_ErrorLogID Primary key (clustered) constraint

#### Used By

	Name
█	dbo.ErrorLog
⚙	dbo.uspLogError

## 8.9.2. Procedures

### 8.9.2.1. Procedure: dbo.usp.LogError

Logs error information in the ErrorLog table about the error that caused execution to jump to the CATCH block of a TRY...CATCH construct. Should be executed from within the scope of a CATCH block otherwise it will return without inserting error information.

#### Input/Output

	Name	Data type	Description
*@	ErrorLogID	int	Output parameter for the stored procedure usp.LogError. Contains the ErrorLogID value corresponding to the row inserted by usp.LogError in the ErrorLog table.

#### Uses

	Name
⚙️	dbo.usp.LogError
📄	dbo.ErrorLog
⚙️	dbo.uspPrintError

#### Used By

	Name
⚙️	dbo.usp.LogError
⚙️	HumanResources.uspUpdateEmployeeHireInfo
⚙️	HumanResources.uspUpdateEmployeeLogin
⚙️	HumanResources.uspUpdateEmployeePersonalInfo
⚡	Production.iWorkOrder
⚡	Production.uWorkOrder
⚡	Purchasing.dVendor
⚡	Purchasing.iPurchaseOrderDetail
⚡	Purchasing.uPurchaseOrderDetail
⚡	Purchasing.uPurchaseOrderHeader
⚡	Sales.iduSalesOrderDetail
⚡	Sales.uSalesOrderHeader

## Script

```
-- uspLogError logs error information in the ErrorLog table about the
-- error that caused execution to jump to the CATCH block of a
-- TRY...CATCH construct. This should be executed from within the scope
-- of a CATCH block otherwise it will return without inserting error
-- information.
CREATE PROCEDURE [dbo].[uspLogError]
    @ErrorLogID [int] = 0 OUTPUT -- contains the ErrorLogID of the row inserted
AS
BEGIN
    SET NOCOUNT ON;
    -- Output parameter value of 0 indicates that error
    -- information was not logged
    SET @ErrorLogID = 0;
    BEGIN TRY
        -- Return if there is no error information to log
        IF ERROR_NUMBER() IS NULL
            RETURN;
        -- Return if inside an uncommittable transaction.
        -- Data insertion/modification is not allowed when
        -- a transaction is in an uncommittable state.
        IF XACT_STATE() = -1
            BEGIN
                PRINT 'Cannot log error since the current transaction is in an uncommittable state. '
                    + 'Rollback the transaction before executing uspLogError in order to successfully log error information.';
                RETURN;
            END
        INSERT [dbo].[ErrorLog]
        (
            [UserName],
            [ErrorNumber],
            [ErrorSeverity],
            [ErrorState],
            [ErrorProcedure],
            [ErrorLine],
            [ErrorMessage]
        )
        VALUES
        (
            CONVERT(sysname, CURRENT_USER),
            ERROR_NUMBER(),
            ERROR_SEVERITY(),
            ERROR_STATE(),
            ERROR_PROCEDURE(),
            ERROR_LINE(),
            ERROR_MESSAGE()
        );
        -- Pass back the ErrorLogID of the row inserted
        SET @ErrorLogID = @@IDENTITY;
    END TRY
    BEGIN CATCH
        PRINT 'An error occurred in stored procedure uspLogError: ';
        EXECUTE [dbo].[uspPrintError];
        RETURN -1;
    END CATCH
END;
```

### 8.9.2.2. Procedure: dbo.uspPrintError

Prints error information about the error that caused execution to jump to the CATCH block of a TRY...CATCH construct. Should be executed from within the scope of a CATCH block otherwise it will return without printing any error information.

#### Used By

Name
dbo.uspPrintError
dbo.usp.LogError
Production.iWorkOrder
Production.uWorkOrder
Purchasing.dVendor
Purchasing.iPurchaseOrderDetail
Purchasing.uPurchaseOrderDetail
Purchasing.uPurchaseOrderHeader
Sales.iduSalesOrderDetail
Sales.uSalesOrderHeader

#### Script

```
-- uspPrintError prints error information about the error that caused
-- execution to jump to the CATCH block of a TRY...CATCH construct.
-- Should be executed from within the scope of a CATCH block otherwise
-- it will return without printing any error information.
CREATE PROCEDURE [dbo].[uspPrintError]
AS
BEGIN
    SET NOCOUNT ON;
    -- Print error information.
    PRINT 'Error ' + CONVERT(varchar(50), ERROR_NUMBER()) +
        ', Severity ' + CONVERT(varchar(5), ERROR_SEVERITY()) +
        ', State ' + CONVERT(varchar(5), ERROR_STATE()) +
        ', Procedure ' + ISNULL(ERROR_PROCEDURE(), '-') +
        ', Line ' + CONVERT(varchar(5), ERROR_LINE());
    PRINT ERROR_MESSAGE();
END;
```

TRIAL

## 9. Sample SSAS Tabular Model

This is a documentation of a sample Adventure Works SSAS tabular Model.

And this is a random Dataedo Data Cartoon to cheer you up:



You can find more cartoons over here: <https://dataedo.com/cartoon>.

You can remove this documentation in your production environment.

## 9.1. Internet Sales



## 9.1.1. Tables

### 9.1.1.1. Table: Customer

#### Columns

	Name	Data type	Description / Attributes
1	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	<b>Identity / Auto increment</b>
2	Customer Id	Int64	<b>Nullable</b>
3	Geography Id	Int64	<b>Nullable</b> References: Geography
4	Customer Alternate Id	String	<b>Nullable</b>
5	Title	String	A courtesy title. For example, Mr. or Ms. <b>Nullable</b>
6	First Name	String	First name of the person. <b>Nullable</b>
7	Middle Name	String	Middle name or middle initial of the person. <b>Nullable</b>
8	Last Name	String	LastName <b>Nullable</b>
9	Name Style	Boolean	0 = The data in FirstName and LastName are stored in western style (first name, last name) order. 1 = Eastern style (last name, first name) order. <b>Nullable</b>
10	Birth Date	DateTime	Date of birth. <b>Nullable</b>
11	Suffix	String	Surname suffix. For example, Sr. or Jr. <b>Nullable</b>
12	Gender	String	<b>Nullable</b>
13	Email Address	String	E-mail address for the person. <b>Nullable</b>
14	Yearly Income	Decimal	<b>Nullable</b>
15	Total Children	Int64	<b>Nullable</b>
16	Number of Children At Home	Int64	<b>Nullable</b>
17	Education	String	<b>Nullable</b>
18	Occupation	String	<b>Nullable</b>
19	Owns House	String	<b>Nullable</b>
20	Number of Cars Owned	Int64	<b>Nullable</b>
21	Address Line 1	String	First street address line. <b>Nullable</b>
22	Address Line 2	String	Second street address line. <b>Nullable</b>
23	Date of First Purchase	DateTime	<b>Nullable</b>
24	Commute Distance	String	<b>Nullable</b>
25	Marital Status	String	<b>Nullable</b>
26	Phone Number	String	<b>Nullable</b>

## Links to

Table	Join	Title / Name / Description
Geography	CustomerGeography Id = GeographyGeography Id	Customer.Geography Id_Geography.Geography Id

## Linked from

Table	Join	Title / Name / Description
Internet Sales	CustomerCustomer Id = Internet SalesCustomer Id	Internet Sales.Customer Id_Customer.Customer Id

## Unique keys

Columns	Name / Description
RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61

## Uses

Name
Customer
Geography

## Used By

Name
Customer
Internet Sales

## 9.1.1.2. Table: Date

### Columns

	Name	Data type	Description / Attributes
■	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	
■	Date	DateTime	Nullable Identity / Auto increment
■	Day Number of Week	Int64	Number of day in a week, where: 1 - Sunday 2 - Monday .... Nullable
■	Day Name	String	Name of a day in a week (e.g. Sunday) Nullable
■	Day of Month	Int64	Nullable
■	Day of Year	Int64	Nullable
■	Week Number of Year	Int64	Nullable
■	Month Name	String	Nullable
■	Month	Int64	Nullable
■	Calendar Quarter	Int64	Calendar quarter of a year (one quarter = three months) Nullable
■	Calendar Year	Int64	Nullable
■	Calendar Semester	Int64	Nullable
■	Fiscal Quarter	Int64	Fiscal quarter of a year where: Fiscal quarter - Calendar quarter 1 - 3 2 - 4 3 - 1 4 - 2 Nullable
■	Fiscal Year	Int64	Fiscal year where first month is July Nullable
■	Fiscal Semester	Int64	Nullable
■	Month Calendar	String	Nullable
■	Day of Week	String	Nullable
■	Days Current Quarter to Date	Int64	Nullable
■	Days in Current Quarter	Int64	Nullable

### Linked from

	Table	Join	Title / Name / Description
→	Internet Sales	DateDate = Internet SalesOrder Date	Internet Sales.Order Date_Date.Date
→	Internet Sales	DateDate = Internet SalesDue Date	Internet Sales.Due Date_Date.Date
→	Internet Sales	DateDate = Internet SalesShip Date	Internet Sales.Ship Date_Date.Date

Used By

Name
Date
→ Internet Sales
→ Internet Sales
→ Internet Sales

TRIAL

### 9.1.1.3. Table: Geography

#### Columns

	Name	Data type	Description / Attributes
█	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	Identity / Auto increment
█	Geography Id	Int64	Nullable
█	City	String	City name Nullable
█	State Province Code	String	State province code appropriate for a country. Nullable
█	State Province Name	String	State province name. Nullable
█	Country Region Code	String	Two letters country code (e.g. AU - Australia) Nullable
█	Country Region Name	String	Country name Nullable
█	Postal Code	String	Nullable
█	Sales Territory Id	Int64	Nullable

#### Linked from

	Table	Join	Title / Name / Description
→	Customer	Geography Geography Id = CustomerGeography Id	Customer.Geography Id_Geography.Geography Id

#### Unique keys

	Columns	Name / Description
█	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61

#### Used By

	Name
█ Geography	
→ Customer	

#### 9.1.1.4. Table: Internet Sales

##### Columns

	Name	Data type	Description / Attributes
💡	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	Identity / Auto increment
☰	Product Id	Int64	Nullable References: Product
☰	Customer Id	Int64	Nullable References: Customer
☰	Promotion Id	Int64	Nullable
☰	Currency Id	Int64	Nullable
☰	Sales Territory Id	Int64	Nullable
☰	Sales Order Number	String	Order number Nullable
☰	Sales Order Line Number	Int64	Order line number Nullable
☰	Revision Number	Int64	Nullable
☰	Order Quantity	Int64	Number of products in order Nullable
☰	Unit Price	Decimal	Nullable
☰	Extended Amount	Decimal	Nullable
☰	Unit Price Discount Pct	Double	Nullable
☰	Discount Amount	Double	Amount of discount in USD (if any) Nullable
☰	Product Standard Cost	Decimal	Standard cost of products in USD Nullable
☰	Total Product Cost	Decimal	Product price with all cost included in USD Nullable
☰	Sales Amount	Decimal	Nullable
☰	Tax Amt	Decimal	Nullable
☰	Freight	Decimal	Nullable
☰	Carrier Tracking Number	String	Nullable
☰	Customer PO Number	String	Nullable
☰	Order Date	DateTime	Nullable References: Date
☰	Due Date	DateTime	Nullable References: Date
☰	Ship Date	DateTime	Nullable References: Date
📊	Margin	Decimal	Nullable
📊	Internet Distinct Count Sales Order	Int64	Nullable
📊	Internet Order Lines Count	Int64	Nullable
📊	Internet Total Units	Int64	Nullable
📊	Internet Total Discount Amount	Double	Nullable

	Name	Data type	Description / Attributes
■	Internet Total Product Cost	Decimal	Nullable
■	Internet Total Sales	Decimal	Nullable
■	Internet Total Margin	Decimal	Nullable
■	Internet Total Tax Amt	Decimal	Nullable
■	Internet Total Freight	Decimal	Nullable
■	Internet Previous Quarter Margin	Decimal	Nullable
■	Internet Current Quarter Margin	Decimal	Nullable
■	Internet Previous Quarter Margin Proportion to QTD	Decimal	Nullable
■	Internet Previous Quarter Sales	Decimal	Nullable
■	Internet Current Quarter Sales	Decimal	Nullable
■	Internet Previous Quarter Sales Proportion to QTD	Decimal	Nullable
⌚	Internet Current Quarter Sales Performance	Double	Nullable
⌚	Internet Current Quarter Margin Performance	Double	Nullable

## Links to

	Table	Join	Title / Name / Description
→	Customer	Internet SalesCustomer Id = CustomerCustomer Id	Internet Sales.Customer Id_Customer.Customer Id
→	Date	Internet SalesOrder Date = DateDate	Internet Sales.Order Date_Date.Date
→	Date	Internet SalesDue Date = DateDate	Internet Sales.Due Date_Date.Date
→	Date	Internet SalesShip Date = DateDate	Internet Sales.Ship Date_Date.Date
→	Product	Internet SalesProduct Id = ProductProduct Id	Internet Sales.Product Id_Product.Product Id

## Unique keys

	Columns	Name / Description
🔑	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61

## Uses

	Name
grid	Internet Sales
→	Customer
→	Date
→	Date
→	Date
→	Product

## 9.1.1.5. Table: Product

### Columns

	Name	Data type	Description / Attributes
1	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	Identity / Auto increment
2	Product Id	Int64	Nullable
3	Product Alternate Id	String	Alternate ID for product Nullable
4	Product Subcategory Id	Int64	Nullable References: Product Subcategory
5	Weight Unit Code	String	Code of a weight unit (LB - pounds, KG, kilograms) Nullable
6	Size Unit Code	String	Code of a size unit (M - meters, CM - centimeters etc.) Nullable
7	Product Name	String	Nullable
8	Standard Cost	Decimal	Nullable
9	Is Finished Product	Boolean	Nullable
10	Color	String	Nullable
11	Safety Stock Level	Int64	Nullable
12	Reorder Point	Int64	Nullable
13	List Price	Decimal	Nullable
14	Size	String	Size in size unit Nullable
15	Size Range	String	Nullable
16	Weight	Double	Weight in weight unit Nullable
17	Days To Manufacture	Int64	Nullable
18	Product Line	String	Nullable
19	Dealer Price	Decimal	Nullable
20	Class	String	Nullable
21	Style	String	Nullable
22	Model Name	String	Nullable
23	Description	String	Nullable
24	Product Start Date	DateTime	Nullable
25	Product End Date	DateTime	Nullable
26	Product Status	String	Nullable
27	Product Subcategory Name	String	Nullable
28	Product Category Name	String	Nullable

## Links to

	Table	Join	Title / Name / Description
→	Product Subcategory	<b>Product</b> Product Subcategory Id = Product Subcategory Product Subcategory Id	Product.Product Subcategory Id_Product Subcategory.Product Subcategory Id

## Linked from

	Table	Join	Title / Name / Description
←	Internet Sales	<b>Product</b> Product Id = Internet Sales Product Product Id	Internet Sales.Product Id_Product.Product Id

## Unique keys

	Columns	Name / Description
🔑	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61

## Uses

	Name
grid	<b>Product</b>
→	Product Subcategory

## Used By

	Name
grid	<b>Product</b>
←	Internet Sales

## 9.1.1.6. Table: Product Category

### Columns

		Name	Data type	Description / Attributes
█	█	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	Identity / Auto increment
█		Product Category Id	Int64	Product category ID Nullable
█		Product Category Alternate Id	Int64	Alternate ID for product category Nullable
█		Product Category Name	String	Product category name Nullable

### Linked from

Table		Join	Title / Name / Description
→	Product Subcategory	Product Category Product Category Id = Product Subcategory.Product Category Id	Product Subcategory.Product Category Id_Product Category.Product Category Id

### Unique keys

Columns		Name / Description
█	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61

### Used By

Name	
█	Product Category
→	Product Subcategory

## 9.1.1.7. Table: Product Subcategory

### Columns

		Name	Data type	Description / Attributes
█	💡	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	Int64	Identity / Auto increment
█		Product Subcategory Id	Int64	Product subcategory ID <b>Nullable</b>
█		Product Subcategory Alternate Id	Int64	Alternate ID for product subcategory <b>Nullable</b>
█		Product Subcategory Name	String	Name for product subcategory <b>Nullable</b>
█		Product Category Id	Int64	Product category ID (references Product Category) <b>Nullable</b> References: Product Category

### Links to

Table		Join	Title / Name / Description
→	Product Category	Product SubcategoryProduct Category Id = Product CategoryProduct Category Id	Product Subcategory.Product Category Id_Product Category.Product Category Id

### Linked from

Table		Join	Title / Name / Description
←	Product	Product SubcategoryProduct Subcategory Id = ProductProduct Subcategory Id	Product.Product Subcategory Id_Product Subcategory.Product Subcategory Id

### Unique keys

Columns		Name / Description
█	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61	RowNumber-2662979B-1795-4F74-8F37-6A1BA8059B61

### Uses

Name	
█	Product Subcategory
→	Product Category

### Used By

Name	
█	Product Subcategory
←	Product

TRIAL