pears.Person

Generated schema reference. Regenerate this page from the SQL unload; keep hand-maintained business notes in the narrative namespace.

Description

Main Person records (includes Comapny Contacts).

Columns

Column	Type	Null	Default	Comment
personid	char(20)	NOT NULL		
staffid	char(20)	NULL		
name	char(30)	NOT NULL		
addr1	char(40)	NULL		
addr2	char(40)	NULL		
addr3	char(40)	NULL		
town	char(30)	NULL		
county	char(30)	NULL		
country	char(30)	NULL		
postcode	char(30)	NULL		
ni	char(15)	NULL		
maritalstatus	char(1)	NULL		
children	smallint	NULL		
payrollnumber	char(15)	NULL		
selfemployed	smallint	NULL	0	
sex	char(1)	NULL		
dob	date	NULL		
available	char(1)	NULL		
notes	long varchar	NULL		
othernotes	long varchar	NULL		
keyname	char(60)	NOT NULL		
status	char(1)	NULL		
noticeperiod	integer	NULL	0	
availdate	date	NULL		
forenames	char(30)	NULL		
surname	char(30)	NULL		

Column	Type	Null	Default	Comment
salutation	char(40)	NULL		
cvtext	long varchar	NULL		
appnumber	char(12)	NULL		
registrationdate	date	NULL		
changedate	timestamp	NULL		
primarycontact	smallint	NULL	0	
alert	char(100)	NULL		
wpdoc	char(10)	NULL		
ExtendedType	char(1)	NULL		
RequirementsMandatory	tinyint	NULL		
OnlyMatchIfKnownAvailable	tinyint	NULL		
PayrollIdentifier	char(1)	NULL		For external payroll link
CVLoc	char(250)	NULL		
CheckAvailabilityDate	date	NULL		A diary date when next to be contacted to determine availability
divisionid	char(20)	NULL		
source	char(1)	NULL		Made available in V2. Hidden by default
ExtraNotes	long varchar	NULL		
TransferNotes	long varchar	NULL		
AvailabilityNotifyDate	date	NULL		Date availability last notified
accordpayrollnumber	char(15)	NULL		
lastcontactevent	timestamp	NULL		Added V2.2.2.13 Will be null for existing people
titleforpayroll	char(20)	NULL		
payrollemailaddress	char(250)	NULL		
directlyemployed	smallint	NULL	0	
PersonWarning	long varchar	NULL		
UnsubscribeToMarketing	tinyint	NULL	0	
RecipientIdentifier	char(50)	NULL		
ExclusiveVacancyID	char(20)	NULL		
compliancestaffid	char(20)	NULL		
CreatedDate	date	NULL	current date	
OriginID	char(20)	NULL		This is the new Source
FilteredLastContactEvent	timestamp	NULL		Like LastContactEvent but for selected contact event types
staffid2	char(20)	NULL		Pulse only field but needed in dast wizard
pdfcv	tinyint	NULL	0	pdf=1, doc=0

Column	Type	Null	Default	Comment
P46Declaration	smallint	NULL	0	0=empty space, 1=First Job since April, no benefits, 2=Had another Job or benefits since April, 3=Have another Job or Pension, 4=Put on OT W1/M1
StudentLoan	smallint	NULL	0	0=No, 1=Yes, 2=empty space
StudentLoanType	smallint	NULL	0	0=empty space, 1=Plan 1 (Northern Ireland, England, Wales), 2=Plan 2 (England and Wales), 3=Plan 4 (Scotland), 4=Postgraduate Loan (England and Wales)

Primary Key

- personid

Foreign Keys

Constraint	Columns	References	Delete/update action
staff	staffid	pears.staff (staffid)	
division	divisionid	pears.Division (divisionid)	ON DELETE SET NULL
vacancyclass	source	pears.vacancyclass (classcode)	
origin	OriginID	pears.Origin (OriginID)	

Referenced By

Table	Constraint	Columns	Referenced columns
pears.AutoMatchNotificationQueue	person	PersonID	personid
pears.AutoMatchPersonConfig	person	PersonID	personid
pears.AutoMatchPersonPreferredShiftType	Person	PersonID	personid
pears.AWRJobMaster	Person	PersonID	personid
pears.AWRWeeklyDetail	Person	PersonID	personid
pears.BroadbeanCandidate	person	personid	personid
pears.CISCard	Person	PersonID	personid
pears.CompanySDS	person	PersonID	personid
pears.CompliancePersonDomain	Person	PersonID	personid
pears.ConsentPersonOption	Person	PersonID	personid
pears.ConsentPersonOptionHistory	Person	PersonID	personid
pears.contactevent	person	personid	personid
pears.diary	person	personid	personid
pears.DocPackValidation	Person	PersonID	personid
pears.EBTimeSheet	Person	PersonID	personid

Table	Constraint	Columns	Referenced columns
pears.employment	person	personid	personid
pears.externalemployment	person	personid	personid
pears.InterPersonLink	Person1	Person1ID	personid
pears.InterPersonLink	Person2	Person2ID	personid
pears.IQXNetCandidateShifts	Person	PersonID	personid
pears.IQXNetUserLink	Person	PersonID	personid
pears.IQXNetYoti	Person	PersonID	personid
pears.MasterRosterFailLog	person	PersonID	personid
pears.MasterRosterPreferredPerson	person	PersonID	personid
pears.MasterRosterShift	Person	DefaultPersonID	personid
pears.NotificationDeepLink	Person	PersonID	personid
pears.OffLimits	Person	PersonID	personid
pears.Pay_Employee	person	PersonID	personid
pears.Pay_Employment	person	PersonID	personid
pears.PayrollDocument	Person	PersonID	personid
pears.PersonIncompatibility	Person1	Person1ID	personid
pears.PersonIncompatibility	Person2	Person2ID	personid
pears.PersonInterest	PersonInterest1	PersonID	personid
pears.PersonShiftPreference	Person	personid	personid
pears.PersonStaffLink	Person	PersonID	personid
pears.PersonWillingness	person	PersonID	personid
pears.PersonWillingnessHistory	person	PersonID	personid
pears.progress	person	personid	personid
pears.psHealthPerson	Person	personid	personid
pears.ReferenceRequest	Person	PersonID	personid
pears.search	person	personid	personid
pears.SentCVs	person	personid	personid
pears.StaffHistory	Person	PersonID	personid
pears.StatusHistory	Person	PersonID	personid
pears.TempPoolMember	Person	PersonID	personid
pears.TempProvTimeSheet	Person	PersonID	personid
pears.TempShift	Person	PersonID	personid
pears.TempShiftProgress	Person	PersonID	personid
pears.TempSubsidy	Person	PersonID	personid
pears.TempTimeSheet	Person	PersonID	personid
pears.TrengoMessage	Person	PersonID	personid
pears.TrengoSent	Person	Personid	personid
pears.TSImage	person	PersonID	personid
pears.TSQueryLog	Person	PersonID	personid
pears.UTRCheck	Person	personID	personid
pears.vacancy	person	RecipientID	personid
pears.WithHolds	person	PersonID	personid

Indexes

Name	Type	Columns	Detail
person_keyname	Index	keyname	
person_appnumber	Index	appnumber	
person_status	Index	status, keyname	
person_postcode	Index	postcode	
forename	Index	forenames	
person_payrollnumber	Index	payrollnumber	
person_lastcontactevent	Index	lastcontactevent	
person_exclusivevacancy	Index	ExclusiveVacancyID	
person_filteredlastcontactevent	Index	FilteredLastContactEvent	
person_ni	Index	ni	
personcvtext	Text index	cvtext	CONFIGURATION “SYS”.“default_char” IMMEDIATE REFRESH
person_notestext	Text index	notes, othernotes	CONFIGURATION “SYS”.“default_char” IMMEDIATE REFRESH

Triggers

Name	Timing	Event
personupdate	after	update of “name”, “addr1”,“addr2”,“addr3”,“town”,“county”,“country”,“postcode”,“ni”,“maritalstatus”,“payrollnumber”,“selfemployed”,“sex”,“dob”,“forenames”,“surname”, “salutation”,“extranotes”,“titleforpayroll” order 1
PersonAudit	before	update of “name”, “addr1”,“addr2”,“addr3”,“town”,“county”,“country”,“postcode”,“ni”,“payrollnumber”,“dob”,“status”,“forenames”,“surname”,“keyname”,“appnumber”,“PayrollIdentifier”,“notes”, “divisionid”,“staffid”,“alert”,“originid”,“titleforpayroll”,“salutation”,“registrationdate”,“directlyemployed”,“UnsubscribeToMarketing”,“personwarning” order 2
PersonUnsubscribe	after	update of “UnsubscribeToMarketing” order 2500
psHealthPersonUpdate	after	update of “forenames”, “surname”,“sex”,“dob”,“addr1”,“addr2”,“addr3”,“town”,“county”,“country”,“titleforpayroll” order 900
Person_InsertTrim	before	insert order 1
Person_UpdateTrim	before	update order 3
InsertStatusPerson	after	insert order 2
UpdateStatusPerson	after	update of “status” order 2
PersonKeyWordsInsert	after	insert order 20
PersonKeyWordsUpdate	after	update of “Salutation”, “Surname”,“Forenames”,“Addr1”,“Addr2”,“Addr3”,“Town”,“County”,“PostCode”,“NI”,“AppNumber”,“PayrollNumber” order 21
PersonDelete	before	delete order 2

Original SQL

```
-- IQX database structure split by table
-- Source: IQXDatabaseStructure - with comments.sql
-- Table: "pears"."Person"
-- Table comment: Main Person records (includes Comapny Contacts).
-- Statement count: 51

CREATE TABLE "pears"."Person" (
    "personid"                      CHAR(20) NOT NULL
   ,"staffid"                       CHAR(20) NULL
   ,"name"                          CHAR(30) NOT NULL
   ,"addr1"                         CHAR(40) NULL
   ,"addr2"                         CHAR(40) NULL
```

```
    ,"addr3"                          CHAR(40) NULL
    ,"town"                           CHAR(30) NULL
    ,"county"                         CHAR(30) NULL
    ,"country"                        CHAR(30) NULL
    ,"postcode"                       CHAR(30) NULL
    ,"ni"                             CHAR(15) NULL
    ,"maritalstatus"                  CHAR(1) NULL
    ,"children"                       SMALLINT NULL
    ,"payrollnumber"                  CHAR(15) NULL
    ,"selfemployed"                   SMALLINT NULL DEFAULT 0
    ,"sex"                            CHAR(1) NULL
    ,"dob"                            DATE NULL
    ,"available"                      CHAR(1) NULL
    ,"notes"                          long VARCHAR NULL
    ,"othernotes"                     long VARCHAR NULL
    ,"keyname"                        CHAR(60) NOT NULL INLINE 60 PREFIX 8
CHECK("length"("trim"("keyname")) > 0)
    ,"status"                         CHAR(1) NULL
    ,"noticeperiod"                   INTEGER NULL DEFAULT 0
    ,"availdate"                      DATE NULL
    ,"forenames"                      CHAR(30) NULL
    ,"surname"                        CHAR(30) NULL
    ,"salutation"                     CHAR(40) NULL
    ,"cvtext"                         long VARCHAR NULL
    ,"appnumber"                      CHAR(12) NULL
    ,"registrationdate"               DATE NULL
    ,"changedate"                     TIMESTAMP NULL
    ,"primarycontact"                 SMALLINT NULL DEFAULT 0
    ,"alert"                          CHAR(100) NULL
    ,"wpdoc"                          CHAR(10) NULL
    ,"ExtendedType"                   CHAR(1) NULL
    ,"RequirementsMandatory"          tinyint NULL
    ,"OnlyMatchIfKnownAvailable"      tinyint NULL
    ,"PayrollIdentifier"              CHAR(1) NULL
    ,"CVLoc"                          CHAR(250) NULL
    ,"CheckAvailabilityDate"          DATE NULL
    ,"divisionid"                     CHAR(20) NULL
    ,"source"                         CHAR(1) NULL
    ,"ExtraNotes"                     long VARCHAR NULL
    ,"TransferNotes"                  long VARCHAR NULL
    ,"AvailabilityNotifyDate"         DATE NULL
    ,"accordpayrollnumber"            CHAR(15) NULL
    ,"lastcontactevent"               TIMESTAMP NULL
    ,"titleforpayroll"                CHAR(20) NULL
    ,"payrollemailaddress"            CHAR(250) NULL
    ,"directlyemployed"               SMALLINT NULL DEFAULT 0
    ,"PersonWarning"                  long VARCHAR NULL
    ,"UnsubscribeToMarketing"         tinyint NULL DEFAULT 0
```

```
    ,"RecipientIdentifier"            CHAR(50) NULL
    ,"ExclusiveVacancyID"             CHAR(20) NULL
    ,"compliancestaffid"              CHAR(20) NULL
    ,"CreatedDate"                    DATE NULL DEFAULT CURRENT DATE
    ,"OriginID"                       CHAR(20) NULL
    ,"FilteredLastContactEvent"       TIMESTAMP NULL
    ,"staffid2"                       CHAR(20) NULL
    ,"pdfcv"                          tinyint NULL DEFAULT 0
    ,"P46Declaration"                 SMALLINT NULL DEFAULT 0
    ,"StudentLoan"                    SMALLINT NULL DEFAULT 0
    ,"StudentLoanType"                SMALLINT NULL DEFAULT 0
    ,PRIMARY KEY ("personid" ASC)
)
GO


COMMENT ON COLUMN "pears"."Person"."PayrollIdentifier" IS
    'For external payroll link'
GO


COMMENT ON COLUMN "pears"."Person"."CheckAvailabilityDate" IS
    'A diary date when next to be contacted to determine availability'
GO


COMMENT ON COLUMN "pears"."Person"."source" IS
    'Made available in V2. Hidden by default'
GO


COMMENT ON COLUMN "pears"."Person"."AvailabilityNotifyDate" IS
    'Date availability last notified'
GO


COMMENT ON COLUMN "pears"."Person"."lastcontactevent" IS
    'Added V2.2.2.13 Will be null for existing people'
GO


COMMENT ON COLUMN "pears"."Person"."OriginID" IS
    'This is the new Source'
GO


COMMENT ON COLUMN "pears"."Person"."FilteredLastContactEvent" IS
    'Like LastContactEvent but for selected contact event types'
```

```
GO


COMMENT ON COLUMN "pears"."Person"."staffid2" IS
    'Pulse only field but needed in dast wizard'
GO


COMMENT ON COLUMN "pears"."Person"."pdfcv" IS
    'pdf=1, doc=0'
GO


COMMENT ON COLUMN "pears"."Person"."P46Declaration" IS
    '0=empty space, 1=First Job since April, no benefits, 2=Had another Job
or benefits since April, 3=Have another Job or Pension, 4=Put on OT W1/M1'
GO


COMMENT ON COLUMN "pears"."Person"."StudentLoan" IS
    '0=No, 1=Yes, 2=empty space'
GO


COMMENT ON COLUMN "pears"."Person"."StudentLoanType" IS
    '0=empty space, 1=Plan 1 (Northern Ireland, England, Wales), 2=Plan 2
(England and Wales), 3=Plan 4 (Scotland), 4=Postgraduate Loan (England and
Wales)'
GO


COMMENT ON TABLE "pears"."Person" IS
    'Main Person records (includes Comapny Contacts).'
GO


ALTER TABLE "pears"."Person"
    ADD FOREIGN KEY "staff" ("staffid" ASC)
    REFERENCES "pears"."staff" ("staffid")
GO


ALTER TABLE "pears"."Person"
    ADD FOREIGN KEY "division" ("divisionid" ASC)
    REFERENCES "pears"."Division" ("divisionid")
    ON DELETE SET NULL
GO
```

```
ALTER TABLE "pears"."Person"
    ADD FOREIGN KEY "vacancyclass" ("source" ASC)
    REFERENCES "pears"."vacancyclass" ("classcode")
GO


ALTER TABLE "pears"."Person"
    ADD FOREIGN KEY "origin" ("OriginID" ASC)
    REFERENCES "pears"."Origin" ("OriginID")
GO


CREATE INDEX "person_keyname" ON "pears"."Person"
    ( "keyname" )
GO


CREATE INDEX "person_appnumber" ON "pears"."Person"
    ( "appnumber" )
GO


CREATE INDEX "person_status" ON "pears"."Person"
    ( "status","keyname" )
GO


CREATE INDEX "person_postcode" ON "pears"."Person"
    ( "postcode" )
GO


CREATE INDEX "forename" ON "pears"."Person"
    ( "forenames" )
GO


CREATE INDEX "person_payrollnumber" ON "pears"."Person"
    ( "payrollnumber" )
GO


CREATE INDEX "person_lastcontactevent" ON "pears"."Person"
    ( "lastcontactevent" DESC )
GO
```

```
CREATE INDEX "person_exclusivevacancy" ON "pears"."Person"
    ( "ExclusiveVacancyID" )
GO


CREATE INDEX "person_filteredlastcontactevent" ON "pears"."Person"
    ( "FilteredLastContactEvent" DESC )
GO


CREATE INDEX "person_ni" ON "pears"."Person"
    ( "ni" )
GO


CREATE TEXT INDEX "personcvtext" ON "pears"."Person"
    ( "cvtext" ) CONFIGURATION "SYS"."default_char" IMMEDIATE REFRESH
GO


CREATE TEXT INDEX "person_notestext" ON "pears"."Person"
    ( "notes","othernotes" ) CONFIGURATION "SYS"."default_char" IMMEDIATE
REFRESH
GO


CREATE TRIGGER "personupdate" after UPDATE OF "name",
"addr1","addr2","addr3","town","county","country","postcode","ni","maritalst
atus","payrollnumber","selfemployed","sex","dob","forenames","surname",
"salutation","extranotes","titleforpayroll" ORDER 1 ON "pears"."Person"
REFERENCING OLD AS "old_pers"
FOR each ROW
BEGIN
  UPDATE "pay_employee" SET "transferbatch" = 0 WHERE "personid" =
"old_pers"."personid" AND "transferbatch" > 0;
  UPDATE "accordemployee" SET "transferbatch" = 0 WHERE "personid" =
"old_pers"."personid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."personupdate" IS
{CREATE TRIGGER personupdate
 after UPDATE OF
name,addr1,addr2,addr3,town,county,country,postcode,ni,maritalstatus,payroll
number,selfemployed,sex,dob,forenames,surname,
salutation,extranotes, titleforpayroll ORDER 1 ON pears.Person
REFERENCING OLD AS old_pers
```

```
FOR each ROW
BEGIN
  UPDATE pay_employee SET transferbatch = 0 WHERE personid =
old_pers.personid AND transferbatch > 0;
  UPDATE accordemployee SET transferbatch = 0 WHERE personid =
old_pers.personid
END
}
GO


CREATE TRIGGER "PersonAudit" BEFORE UPDATE OF "name",
"addr1","addr2","addr3","town","county","country","postcode","ni","payrollnu
mber","dob","status","forenames","surname","keyname","appnumber","PayrollIde
ntifier","notes",
"divisionid","staffid","alert","originid","titleforpayroll","salutation","re
gistrationdate","directlyemployed","UnsubscribeToMarketing","personwarning"
ORDER 2 ON "pears"."Person"
REFERENCING OLD AS "old_pers" NEW AS "new_pers"
FOR each ROW
WHEN(EXISTS(SELECT * FROM "AuditItems" WHERE "AreaName" = 'Person' AND
"AuditFlag" = 1))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @OldDescrip CHAR(250);
  DECLARE @NewDescrip CHAR(250);
  DECLARE @PersonName CHAR(250);
  SELECT "string"(',',"list"("ItemName"),',') INTO @AuditList FROM
"AuditItems" WHERE "AreaName" = 'Person' AND "AuditFlag" = 1;
  SET @PersonName = "old_pers"."Name";
  -- DOB
  IF "locate"(@AuditList,',DOB,') > 0 AND UPDATE("dob") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Date of Birth
Updated - ',@PersonName),"old_pers"."dob","new_pers"."dob")
  END IF;
  IF "locate"(@AuditList,',Title for Payroll,') > 0 AND
UPDATE("titleforpayroll") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Title for
Payroll Updated -
',@PersonName),"old_pers"."titleforpayroll","new_pers"."titleforpayroll")
  END IF;
  -- NI Number
  IF "locate"(@AuditList,',National Insurance Number,') > 0 AND UPDATE("NI")
THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('NI Updated -
',@PersonName),"old_pers"."ni","new_pers"."ni")
  END IF;
  IF "locate"(@AuditList,',Warning,') > 0 AND UPDATE("personwarning") THEN
```

```
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Warning Updated
- ',@PersonName),"old_pers"."personwarning","new_pers"."personwarning")
  END IF;
  -- Reg Date
  IF "locate"(@AuditList,',Registration Date,') > 0 AND
UPDATE("RegistrationDate") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Registration
Date Updated -
',@PersonName),"old_pers"."RegistrationDate","new_pers"."RegistrationDate")
  END IF;
  -- Notes
  IF "locate"(@AuditList,',Notes,') > 0 AND UPDATE("Notes") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Notes Updated -
',@PersonName),"old_pers"."notes","new_pers"."notes")
  END IF;
  -- App Number
  IF "locate"(@AuditList,',Registration Number,') > 0 AND
UPDATE("AppNumber") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Registration
Number Updated -
',@PersonName),"old_pers"."AppNumber","new_pers"."AppNumber")
  END IF;
  -- Status
  IF "locate"(@AuditList,',Status,') > 0 AND UPDATE("Status") THEN
    SELECT "name" INTO @OldDescrip FROM "status" WHERE "type" = 'P' AND
"status"."status" = "old_pers"."status";
    SELECT "name" INTO @NewDescrip FROM "status" WHERE "type" = 'P' AND
"status"."status" = "new_pers"."status";
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Status Updated
- ',@PersonName),"string"("old_pers"."Status",' -
',@OldDescrip),"string"("new_pers"."Status",' - ',@NewDescrip))
  END IF;
  -- Payroll Identifier
  IF "locate"(@AuditList,',Payroll Company,') > 0 AND
UPDATE("PayrollIdentifier") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Payroll
Identifier Updated -
',@PersonName),"old_pers"."PayrollIdentifier","new_pers"."PayrollIdentifier"
)
  END IF;
  -- Division
  IF "locate"(@AuditList,',Division,') > 0 AND UPDATE("DivisionID") THEN
    SELECT "name" INTO @OldDescrip FROM "Division" WHERE "divisionid" =
"old_pers"."DivisionID";
    SELECT "name" INTO @NewDescrip FROM "Division" WHERE "divisionid" =
"new_pers"."DivisionID";
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Division
Updated - ',@PersonName),@OldDescrip,@NewDescrip)
```

```
  END IF;
  -- Consultant
  IF "locate"(@AuditList,',Consultant,') > 0 AND UPDATE("StaffID") THEN
    SELECT "userid" INTO @OldDescrip FROM "staff" WHERE "staffid" =
"old_pers"."StaffID";
    SELECT "userid" INTO @NewDescrip FROM "staff" WHERE "staffid" =
"new_pers"."StaffID";
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Consultant
Updated - ',@PersonName),@OldDescrip,@NewDescrip);
    INSERT INTO "staffhistory"(
"staffhistoryid","personid","oldstaff","newstaff","whoentered","whenentered"
) VALUES(
"uniquekey"("new_pers"."StaffID"),"new_pers"."personid","old_pers"."StaffID"
,"new_pers"."StaffID","userstaffid",CURRENT TIMESTAMP )
  END IF;
  -- Address
  IF "locate"(@AuditList,',Address,') > 0 AND(UPDATE("addr1") OR
UPDATE("addr2") OR UPDATE("addr3") OR UPDATE("town") OR UPDATE("county") OR
UPDATE("country") OR UPDATE("postcode")) THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Address Updated
- ',@PersonName),"string"("old_pers"."addr1",', ',"old_pers"."addr2",',
',"old_pers"."addr3",', ',"old_pers"."town",', ',"old_pers"."county",',
',"old_pers"."postcode",',
',"old_pers"."country"),"string"("new_pers"."addr1",',
',"new_pers"."addr2",', ',"new_pers"."addr3",', ',"new_pers"."town",',
',"new_pers"."county",', ',"new_pers"."postcode",', ',"new_pers"."country"))
  END IF;
  -- Alert
  IF "locate"(@AuditList,',Alert,') > 0 AND UPDATE("Alert") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Alert Updated -
',@PersonName),"old_pers"."Alert","new_pers"."Alert")
  END IF;
  -- Name
  IF "locate"(@AuditList,',Name,') > 0 THEN
    IF UPDATE("name") THEN
      CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Name Updated
- ',@PersonName),"old_pers"."name","new_pers"."name")
    END IF;
    -- Forenames
    IF UPDATE("Forenames") THEN
      CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Forenames
Updated - ',@PersonName),"old_pers"."Forenames","new_pers"."Forenames")
    END IF;
    -- Surname
    IF UPDATE("Surname") THEN
      CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Surname
Updated - ',@PersonName),"old_pers"."Surname","new_pers"."Surname")
    END IF;
```

```
    -- Sslutation
    IF UPDATE("Sslutation") THEN
      CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Salutation
Updated - ',@PersonName),"old_pers"."Salutation","new_pers"."Salutation")
    END IF;
    -- Keyname
    IF UPDATE("Keyname") THEN
      CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Keyname
Updated - ',@PersonName),"old_pers"."Keyname","new_pers"."Keyname")
    END IF END IF;
  -- Payroll Number
  IF "locate"(@AuditList,',Payroll Number,') > 0 AND UPDATE("PayrollNumber")
THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Payroll Number
Updated -
',@PersonName),"old_pers"."PayrollNumber","new_pers"."PayrollNumber")
  END IF;
  -- AWR Exempt
  IF "locate"(@AuditList,',AWR Exemption Reason,') > 0 AND
UPDATE("directlyemployed") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('AWR Exemption
Reason Updated -
',@PersonName),"string"("old_pers"."directlyemployed"),"string"("new_pers"."
directlyemployed"))
  END IF;
  -- Source
  IF "locate"(@AuditList,',Source,') > 0 AND UPDATE("originid") THEN
    SELECT "descrip" INTO @OldDescrip FROM "origin" WHERE "originid" =
"old_pers"."originid";
    SELECT "descrip" INTO @NewDescrip FROM "origin" WHERE "originid" =
"new_pers"."originid";
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Source Updated
- ',@PersonName),"string"(@OldDescrip),"string"(@NewDescrip))
  END IF;
  -- Unsubscribe To Marketing
  IF "locate"(@AuditList,',Unsubscribe To Marketing,') > 0 AND
UPDATE("UnsubscribeToMarketing") THEN
    CALL "AuditLog"('PERSON',"old_pers"."personid","string"('Unsubscribe To
Marketing -
',@PersonName),"string"("old_pers"."UnsubscribeToMarketing"),"string"("new_p
ers"."UnsubscribeToMarketing"))
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."PersonAudit" IS
{CREATE TRIGGER PersonAudit
```

```
 BEFORE UPDATE OF
name,addr1,addr2,addr3,town,county,country,postcode,ni,payrollnumber,dob,STA
TUS,forenames,surname,keyname,appnumber,PayrollIdentifier,notes,
divisionid,staffid,alert,originid, titleforpayroll, salutation,
registrationdate, directlyemployed,UnsubscribeToMarketing, personwarning
ORDER 2 ON pears.Person
REFERENCING OLD AS old_pers NEW AS new_pers
FOR each ROW
WHEN(EXISTS(SELECT* FROM AuditItems WHERE AreaName = 'Person' AND AuditFlag
= 1))
BEGIN
  DECLARE @AuditList long VARCHAR;
  DECLARE @OldDescrip CHAR(250);
  DECLARE @NewDescrip CHAR(250);
  DECLARE @PersonName CHAR(250);
  SELECT string(',',list(ItemName),',') INTO @AuditList FROM AuditItems
WHERE AreaName = 'Person' AND AuditFlag = 1;
  SET @PersonName=old_pers.Name;
  -- DOB
  IF locate(@AuditList,',DOB,') > 0 AND UPDATE(dob) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Date of Birth Updated -
',@PersonName),old_pers.dob,new_pers.dob)
  END IF;
  IF locate(@AuditList,',Title for Payroll,') > 0 AND
UPDATE(titleforpayroll) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Title for Payroll
Updated - ',@PersonName),old_pers.titleforpayroll,new_pers.titleforpayroll)
  END IF;
  -- NI Number
  IF locate(@AuditList,',National Insurance Number,') > 0 AND UPDATE(NI)
THEN
    CALL AuditLog('PERSON',old_pers.personid,string('NI Updated -
',@PersonName),old_pers.ni,new_pers.ni)
  END IF;
  IF locate(@AuditList,',Warning,') > 0 AND UPDATE(personwarning) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Warning Updated -
',@PersonName),old_pers.personwarning,new_pers.personwarning)
  END IF;
  -- Reg Date
  IF locate(@AuditList,',Registration Date,') > 0 AND
UPDATE(RegistrationDate) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Registration Date
Updated -
',@PersonName),old_pers.RegistrationDate,new_pers.RegistrationDate)
  END IF;
  -- Notes
  IF locate(@AuditList,',Notes,') > 0 AND UPDATE(Notes) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Notes Updated -
```

```
',@PersonName),old_pers.notes,new_pers.notes)
  END IF;
  -- App Number
  IF locate(@AuditList,',Registration Number,') > 0 AND UPDATE(AppNumber)
THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Registration Number
Updated - ',@PersonName),old_pers.AppNumber,new_pers.AppNumber)
  END IF;
  -- Status
  IF locate(@AuditList,',Status,') > 0 AND UPDATE(STATUS) THEN
    SELECT name INTO @OldDescrip FROM STATUS WHERE TYPE = 'P' AND
STATUS.status = old_pers.status;
    SELECT name INTO @NewDescrip FROM STATUS WHERE TYPE = 'P' AND
STATUS.status = new_pers.status;
    CALL AuditLog('PERSON',old_pers.personid,string('Status Updated -
',@PersonName),string(old_pers.Status,' -
',@OldDescrip),string(new_pers.Status,' - ',@NewDescrip))
  END IF;
  -- Payroll Identifier
  IF locate(@AuditList,',Payroll Company,') > 0 AND
UPDATE(PayrollIdentifier) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Payroll Identifier
Updated -
',@PersonName),old_pers.PayrollIdentifier,new_pers.PayrollIdentifier)
  END IF;
  -- Division
  IF locate(@AuditList,',Division,') > 0 AND UPDATE(DivisionID) THEN
    SELECT name INTO @OldDescrip FROM Division WHERE divisionid =
old_pers.DivisionID;
    SELECT name INTO @NewDescrip FROM Division WHERE divisionid =
new_pers.DivisionID;
    CALL AuditLog('PERSON',old_pers.personid,string('Division Updated -
',@PersonName),@OldDescrip,@NewDescrip)
  END IF;
  -- Consultant
  IF locate(@AuditList,',Consultant,') > 0 AND UPDATE(StaffID) THEN
    SELECT userid INTO @OldDescrip FROM staff WHERE staffid =
old_pers.StaffID;
    SELECT userid INTO @NewDescrip FROM staff WHERE staffid =
new_pers.StaffID;
    CALL AuditLog('PERSON',old_pers.personid,string('Consultant Updated -
',@PersonName),@OldDescrip,@NewDescrip);
    INSERT INTO staffhistory (staffhistoryid, personid, oldstaff, newstaff,
whoentered, whenentered) VALUES
(uniquekey(new_pers.StaffID),new_pers.personid, old_pers.StaffID,
new_pers.StaffID, userstaffid, CURRENT TIMESTAMP)
  END IF;
  -- Address
```

```
  IF locate(@AuditList,',Address,') > 0 AND (UPDATE(addr1) OR UPDATE(addr2)
OR UPDATE(addr3) OR UPDATE(town) OR UPDATE(county) OR UPDATE(country) OR
UPDATE(postcode)) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Address Updated -
',@PersonName),string(old_pers.addr1,', ',old_pers.addr2,',
',old_pers.addr3,', ',old_pers.town,', ',old_pers.county,',
',old_pers.postcode,', ',old_pers.country),string(new_pers.addr1,',
',new_pers.addr2,', ',new_pers.addr3,', ',new_pers.town,',
',new_pers.county,', ',new_pers.postcode,', ',new_pers.country))
  END IF;
  -- Alert
  IF locate(@AuditList,',Alert,') > 0 AND UPDATE(Alert) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Alert Updated -
',@PersonName),old_pers.Alert,new_pers.Alert)
  END IF;
  -- Name
  IF locate(@AuditList,',Name,') > 0 THEN
    IF UPDATE(name) THEN
      CALL AuditLog('PERSON',old_pers.personid,string('Name Updated -
',@PersonName),old_pers.name,new_pers.name)
    END IF;
    -- Forenames
    IF UPDATE(Forenames) THEN
      CALL AuditLog('PERSON',old_pers.personid,string('Forenames Updated -
',@PersonName),old_pers.Forenames,new_pers.Forenames)
    END IF;
    -- Surname
    IF UPDATE(Surname) THEN
      CALL AuditLog('PERSON',old_pers.personid,string('Surname Updated -
',@PersonName),old_pers.Surname,new_pers.Surname)
    END IF;
    -- Sslutation
    IF UPDATE(Sslutation) THEN
      CALL AuditLog('PERSON',old_pers.personid,string('Salutation Updated -
',@PersonName),old_pers.Salutation,new_pers.Salutation)
    END IF;
    -- Keyname
    IF UPDATE(Keyname) THEN
      CALL AuditLog('PERSON',old_pers.personid,string('Keyname Updated -
',@PersonName),old_pers.Keyname,new_pers.Keyname)
    END IF
  END IF;
  -- Payroll Number
  IF locate(@AuditList,',Payroll Number,') > 0 AND UPDATE(PayrollNumber)
THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Payroll Number Updated
- ',@PersonName),old_pers.PayrollNumber,new_pers.PayrollNumber)
  END IF;
```

```
  -- AWR Exempt
  IF locate(@AuditList,',AWR Exemption Reason,') > 0 AND
UPDATE(directlyemployed) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('AWR Exemption Reason
Updated -
',@PersonName),string(old_pers.directlyemployed),string(new_pers.directlyemp
loyed))
  END IF;
  -- Source
  IF locate(@AuditList,',Source,') > 0 AND UPDATE(originid) THEN
    SELECT descrip INTO @OldDescrip FROM origin WHERE originid =
old_pers.originid;
    SELECT descrip INTO @NewDescrip FROM origin WHERE originid =
new_pers.originid;
    CALL AuditLog('PERSON',old_pers.personid,string('Source Updated -
',@PersonName),string(@OldDescrip),string(@NewDescrip))
  END IF;
  -- Unsubscribe To Marketing
  IF locate(@AuditList,',Unsubscribe To Marketing,') > 0 AND
UPDATE(UnsubscribeToMarketing) THEN
    CALL AuditLog('PERSON',old_pers.personid,string('Unsubscribe To
Marketing -
',@PersonName),string(old_pers.UnsubscribeToMarketing),string(new_pers.Unsub
scribeToMarketing))
  END IF;
END
}
GO


CREATE TRIGGER "PersonUnsubscribe" after UPDATE OF "UnsubscribeToMarketing"
ORDER 2500 ON "pears"."Person"
REFERENCING OLD AS "old_name" NEW AS "new_name"
FOR each ROW /* WHEN( search_condition ) */
BEGIN
  DECLARE "emailaddr" CHAR(100);
  DECLARE "existingemail" INTEGER;
  IF "new_name"."UnsubscribeToMarketing" = 1 THEN
    SELECT "isnull"("getphone"('P','E-mail',"old_name"."personid"),'') INTO
"emailaddr";
    SELECT "count"("email") INTO "existingemail" FROM
"UnsubscribeToMarketing" WHERE "email" = "emailaddr";
    IF "emailaddr" <> '' AND "existingemail" = 0 THEN
      INSERT INTO "UnsubscribeToMarketing"( "email" ) VALUES( "emailaddr" )
;
      UPDATE "mailerselectionmember" SET "unsubscribe" = 1 WHERE
"datesubscribed" IS NOT NULL AND "record" = "old_name"."personid" AND
"mailerselectionid" = any(SELECT "mailerselectionid" FROM "mailerselection"
```

```
WHERE "type" = 'P');
      DELETE FROM "mailerselectionmember" WHERE "record" =
"old_name"."personid" AND "datesubscribed" IS NULL AND "mailerselectionid" =
any(SELECT "mailerselectionid" FROM "mailerselection" WHERE "type" = 'P')
    END IF
  ELSE
    DELETE FROM "UnsubscribeToMarketing" WHERE "email" = any(SELECT
"getphone"('P','E-mail',"old_name"."personid"))
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."PersonUnsubscribe"
IS
{CREATE TRIGGER PersonUnsubscribe
 AFTER UPDATE OF UnsubscribeToMarketing
 ORDER 2500 ON Person
  REFERENCING OLD AS old_name NEW AS new_name
 FOR EACH ROW /* WHEN( search_condition ) */
 BEGIN
   DECLARE emailaddr CHAR(100);
   DECLARE existingemail INTEGER;
   IF new_name.UnsubscribeToMarketing = 1 THEN
     SELECT isnull(getphone('P','E-mail', old_name.personid),'') INTO
emailaddr;
     SELECT COUNT(email) INTO existingemail FROM UnsubscribeToMarketing
WHERE email=emailaddr;
     IF emailaddr<>'' AND existingemail=0 THEN
        INSERT INTO UnsubscribeToMarketing (email) VALUES (emailaddr);
        UPDATE mailerselectionmember SET unsubscribe=1 WHERE datesubscribed
IS NOT NULL AND record=old_name.personid AND mailerselectionid IN (SELECT
mailerselectionid FROM mailerselection WHERE TYPE='P') ;
        DELETE FROM mailerselectionmember WHERE record=old_name.personid AND
datesubscribed IS NULL AND mailerselectionid IN (SELECT mailerselectionid
FROM mailerselection WHERE TYPE='P') ;
     END IF;
   ELSE
     DELETE FROM UnsubscribeToMarketing WHERE email IN ( SELECT
getphone('P','E-mail', old_name.personid) )
   END IF;
 END
}
GO


CREATE TRIGGER "psHealthPersonUpdate" after UPDATE OF "forenames",
"surname","sex","dob","addr1","addr2","addr3","town","county","country","tit
```

```
leforpayroll" ORDER 900 ON "pears"."person"
REFERENCING NEW AS "new_rec"
FOR each ROW
BEGIN
  IF "psHealthCanSendPerson"("new_rec"."personid",NULL) = 1 THEN
    CALL "psHealthInsertUpdatePerson"("new_rec"."personid",NULL)
  END IF
END
GO


CREATE TRIGGER "Person_InsertTrim" BEFORE INSERT ORDER 1 ON
"pears"."person"
REFERENCING NEW AS "new_p"
FOR each ROW
BEGIN
  SET "new_p"."keyname" = "trim"("new_p"."keyname");
  SET "new_p"."surname" = "trim"("new_p"."surname");
  SET "new_p"."payrollnumber" = "trim"("new_p"."payrollnumber")
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."Person_InsertTrim"
IS
{CREATE TRIGGER Person_InsertTrim
 BEFORE INSERT ORDER 1 ON
pears.person
REFERENCING NEW AS new_p
FOR each ROW
BEGIN
  SET new_p.keyname = TRIM(new_p.keyname);
  SET new_p.surname = TRIM(new_p.surname);
  SET new_p.payrollnumber = TRIM(new_p.payrollnumber)
END
}
GO


CREATE TRIGGER "Person_UpdateTrim" BEFORE UPDATE ORDER 3 ON
"pears"."person"
REFERENCING NEW AS "new_p"
FOR each ROW
BEGIN
  SET "new_p"."keyname" = "trim"("new_p"."keyname");
  SET "new_p"."surname" = "trim"("new_p"."surname");
  SET "new_p"."payrollnumber" = "trim"("new_p"."payrollnumber")
END
```

```
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."Person_UpdateTrim"
IS
{CREATE TRIGGER Person_UpdateTrim
 BEFORE UPDATE ORDER 3 ON
pears.person
REFERENCING NEW AS new_p
FOR each ROW
BEGIN
  SET new_p.keyname = TRIM(new_p.keyname);
  SET new_p.surname = TRIM(new_p.surname);
  SET new_p.payrollnumber = TRIM(new_p.payrollnumber)
END
}
GO


CREATE TRIGGER "InsertStatusPerson" after INSERT ORDER 2 ON
"pears"."Person"
REFERENCING NEW AS "new_co"
FOR each ROW
BEGIN
  INSERT INTO "StatusHistory"(
"StatusHistoryID","personid","staffid","newstatus" ) VALUES
    (
"uniquekey"("new_co"."personid"),"new_co"."personid","userstaffid","new_co".
"status" )
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."InsertStatusPerson"
IS
{CREATE TRIGGER InsertStatusPerson
 after INSERT ORDER 2 ON
pears.Person
REFERENCING NEW AS new_co
FOR each ROW
BEGIN
  INSERT INTO StatusHistory (StatusHistoryID, personid, staffid, newstatus)
    VALUES
    (uniquekey(new_co.personid),new_co.personid, userstaffid,new_co.status)
END
}
GO
```

```
CREATE TRIGGER "UpdateStatusPerson" after UPDATE OF "status"
ORDER 2 ON "pears"."person"
REFERENCING OLD AS "old_co" NEW AS "new_co"
FOR each ROW
BEGIN
  DECLARE "ID" CHAR(20);
  SELECT FIRST "StatusHistoryID" INTO "ID" FROM "StatusHistory" WHERE
"personid" = "new_co"."personid"
    AND "datediff"("minute","whenentered",CURRENT TIMESTAMP) < 1 AND
"newstatus" = "old_co"."status" AND "oldstatus" = "new_co"."status" ORDER BY
"whenentered" DESC;
  IF "ID" IS NOT NULL THEN
    DELETE FROM "StatusHistory" WHERE "StatusHistoryID" = "ID"
  ELSE
    INSERT INTO "StatusHistory"(
"StatusHistoryID","personid","staffid","oldstatus","newstatus" ) VALUES
      (
"uniquekey"("new_co"."personid"),"new_co"."personid","userstaffid","old_co".
"status","new_co"."status" )
  END IF
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."UpdateStatusPerson"
IS
{CREATE TRIGGER UpdateStatusPerson
 after UPDATE OF STATUS
ORDER 2 ON pears.person
REFERENCING OLD AS old_co NEW AS new_co
FOR each ROW
BEGIN
  DECLARE ID CHAR(20);
   SELECT FIRST StatusHistoryID INTO ID FROM StatusHistory WHERE personid =
new_co.personid
     AND datediff(MINUTE, whenentered, CURRENT TIMESTAMP) < 1 AND newstatus
= old_co.status AND oldstatus = new_co.status ORDER BY whenentered DESC;
   IF ID IS NOT NULL THEN
     DELETE FROM StatusHistory WHERE StatusHistoryID = ID
   ELSE
     INSERT INTO StatusHistory (StatusHistoryID, personid, staffid,
oldstatus, newstatus)
      VALUES
      (uniquekey(new_co.personid),new_co.personid, userstaffid,
old_co.status, new_co.status)
   END IF;
END
```

```
}
GO


CREATE TRIGGER "PersonKeyWordsInsert" after INSERT ORDER 20 ON
"pears"."Person"
REFERENCING NEW AS "NewRow"
FOR each ROW
BEGIN
  INSERT INTO "PersonKeyWords"( "PersonID","RefreshRequired" ) ON existing
UPDATE defaults off VALUES( "NewRow"."PersonID",1 )
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Person"."PersonKeyWordsInsert" IS
{CREATE TRIGGER PersonKeyWordsInsert
after INSERT  ORDER 20 ON
 pears.Person
 REFERENCING NEW AS NewRow
FOR each ROW
BEGIN
   INSERT INTO PersonKeyWords(PersonID,RefreshRequired) ON existing UPDATE
VALUES(NewRow.PersonID,1);
END
}
GO


CREATE TRIGGER "PersonKeyWordsUpdate" after UPDATE OF "Salutation",
"Surname","Forenames","Addr1","Addr2","Addr3","Town","County","PostCode","NI
","AppNumber","PayrollNumber" ORDER 21 ON
"pears"."Person"
REFERENCING NEW AS "NewRow"
FOR each ROW
BEGIN
  UPDATE "PersonKeyWords" SET "RefreshRequired" = 1 WHERE "PersonID" =
"NewRow"."PersonID"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER
"pears"."Person"."PersonKeyWordsUpdate" IS
{CREATE TRIGGER PersonKeyWordsUpdate
after  UPDATE OF
Salutation,Surname,Forenames,Addr1,Addr2,Addr3,Town,County,PostCode,NI,AppNu
```

```
mber,PayrollNumber ORDER 21 ON
   pears.Person
  REFERENCING NEW AS NewRow
 FOR each ROW
 BEGIN
 UPDATE PersonKeyWords SET RefreshRequired = 1 WHERE PersonID =
NewRow.PersonID;
 END
}
GO


CREATE TRIGGER "PersonDelete" BEFORE DELETE ORDER 2 ON
"pears"."person"
REFERENCING OLD AS "old_name"
FOR each ROW
BEGIN
  DELETE FROM "compliancepersonstatus" WHERE "personid" =
"old_name"."personid"
END
GO


COMMENT TO PRESERVE FORMAT ON TRIGGER "pears"."Person"."PersonDelete" IS
{CREATE TRIGGER PersonDelete
 BEFORE DELETE  ORDER 2 ON
pears."person"
REFERENCING OLD AS old_name
FOR each ROW
BEGIN
    DELETE FROM "compliancepersonstatus" WHERE "personid" =
"old_name"."personid"
END
}
GO
```