PRÁCTICA I: Módulo Data 101

Primera parte:

a) Estudio de STG_PRODUCTOS_CRM:

Analisis STG_PRODUCTOS_CRM.sql

```
USE STAGE;

SELECT COUNT(*) TOTAL_REGISTROS
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_ID)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PRODUCT_ID
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_ID)) <> 0 THEN PRODUCT_ID
ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_ID
, SUM(CASE WHEN LENGTH(TRIM(CUSTOMER_ID)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_CUSTOMER_ID
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(CUSTOMER_ID)) <> 0 THEN
CUSTOMER_ID ELSE 0 END) AS TOTAL_DISTINTOS_CUSTOMER_ID
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_NAME)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PRODUCT_NAME
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_NAME)) <> 0 THEN
PRODUCT_NAME ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_NAME
, SUM(CASE WHEN LENGTH(TRIM(ACCESS_POINT)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_ACCESS_POINT
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(ACCESS_POINT)) <> 0 THEN
ACCESS_POINT ELSE 0 END) AS TOTAL_DISTINTOS_ACCESS_POINT
, SUM(CASE WHEN LENGTH(TRIM(CHANNEL)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_CHANNEL
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(CHANNEL)) <> 0 THEN CHANNEL ELSE
0 END) AS TOTAL_DISTINTOS_CHANNEL
, SUM(CASE WHEN LENGTH(TRIM(AGENT_CODE)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_AGENT_CODE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AGENT_CODE)) <> 0 THEN AGENT_CODE
ELSE 0 END) AS TOTAL_DISTINTOS_AGENT_CODE
, SUM(CASE WHEN LENGTH(TRIM(START_DATE)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_START_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(START_DATE)) <> 0 THEN START_DATE
ELSE 0 END) AS TOTAL_DISTINTOS_START_DATE
, SUM(CASE WHEN LENGTH(TRIM(INSTALL_DATE)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_INSTALL_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(INSTALL_DATE)) <> 0 THEN
INSTALL_DATE ELSE 0 END) AS TOTAL_DISTINTOS_INSTALL_DATE
, SUM(CASE WHEN LENGTH(TRIM(END_DATE)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_END_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(END_DATE)) <> 0 THEN END_DATE
ELSE 0 END) AS TOTAL_DISTINTOS_END_DATE
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_CITY)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PRODUCT_CITY
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_CITY)) <> 0 THEN
PRODUCT_CITY ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_CITY
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_ADDRESS)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PRODUCT_ADDRESS
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_ADDRESS)) <> 0 THEN
PRODUCT_ADDRESS ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_ADDRESS
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_POSTAL_CODE)) <> 0 THEN 1 ELSE 0
END) AS TOTAL_PRODUCT_POSTAL_CODE
```

```
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_POSTAL_CODE)) <> 0 THEN
PRODUCT_POSTAL_CODE ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_POSTAL_CODE
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_STATE)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PRODUCT_STATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_STATE)) <> 0 THEN
PRODUCT_STATE ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_STATE
, SUM(CASE WHEN LENGTH(TRIM(PRODUCT_COUNTRY)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PRODUCT_COUNTRY
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PRODUCT_COUNTRY)) <> 0 THEN
PRODUCT_COUNTRY ELSE 0 END) AS TOTAL_DISTINTOS_PRODUCT_COUNTRY
FROM STAGE.STG_PRODUCTOS_CRM;
```

Obtenemos los siguientes resultados:

TOTAL_REGISTROS:	78495
TOTAL_PRODUCT_ID:	78495
TOTAL_DISTINTOS_PRODUCT_ID:	78495
TOTAL_CUSTOMER_ID:	78495
TOTAL_DISTINTOS_CUSTOMER_ID:	8001
TOTAL_PRODUCT_NAME:	78495
TOTAL_DISTINTOS_PRODUCT_NAME:	6
TOTAL_ACCESS_POINT:	78274
TOTAL_DISTINTOS_ACCESS_POINT:	78275
TOTAL_CHANNEL:	78274

TOTAL_DISTINTOS_CHANNEL:	5
TOTAL_AGENT_CODE:	42630
TOTAL_DISTINTOS_AGENT_CODE:	701
TOTAL_START_DATE:	78495
TOTAL_DISTINTOS_START_DATE:	8035
TOTAL_INSTALL_DATE:	75363
TOTAL_DISTINTOS_INSTALL_DATE:	75360
TOTAL_END_DATE:	46684
TOTAL_DISTINTOS_END_DATE:	46683
TOTAL_PRODUCT_CITY:	78274
TOTAL_DISTINTOS_PRODUCT_CITY:	82
TOTAL_PRODUCT_ADDRESS:	78274
TOTAL_DISTINTOS_PRODUCT_ADDRESS:	77037
TOTAL_PRODUCT_POSTAL_CODE:	78274
TOTAL_DISTINTOS_PRODUCT_POSTAL_CODE:	274
TOTAL_PRODUCT_STATE:	78090
TOTAL_DISTINTOS_PRODUCT_STATE:	4
TOTAL_PRODUCT_COUNTRY:	78274
TOTAL_DISTINTOS_PRODUCT_COUNTRY:	2

Conclusiones:

- 1269 (1,6%) campos nulos en el campo INSTALL_DATE y END_DATE, revisando más a fondo los datos nos encontramos con que START_DATE es mayor al año actual (2017) en un total de 1229 casos.
  ¿Qué ha ocurrido con estos servicios contratados? ¿Son bajas que se produjeron antes de realizarse la instalación del servicio (amagos)? ¿Son errores del funcional?
  De los 40 registros que pertenecen a 2017, hay 28 del primer semestre del año.
  Los otros 12 son de los meses de julio, agosto y septiembre.
  ¿Amagos? ¿Errores del funcional?

- Hay 4 registros que no tienen rellenos los campos PRODUCT_COUNTRY, PRODUCT_STATE, PRODUCT_CITY, PRODUCT_ADDRESS y PRODUCT_POSTAL_CODE ¿Errores? Intentaría averiguar la procedencia de estos registros, y ver si podemos corregir estos datos.
- Hay 217 registros que sólo tienen rellenos el campo PRODUCT_STATE. Al igual que los anteriores está claro que existe un fallo a la hora de guardar datos en el operacional que estaría bien contemplar y tomar una decisión que "arregle" lo que pueda estar fallando en el aplicativo.
- A la cuestión de si es interesante guardar la dirección para la instalación en ODS en principio yo sí que la guardaría, ya que nos puede servir, por un lado, para obtener una dirección del cliente en caso de que no haya ninguna en la tabla de clientes y por otro lado podríamos sacar información de tipos de productos contratados por regiones.
- El campo AGENT_CODE está relleno con un código del agente, pero no disponemos de una tabla que relacione código del agente con información de éste. Luego, en la tabla STG_CONTACTOS_IVR aparece también un campo AGENT que está relleno con una cadena de texto que parece coincidir con el nombre del usuario del agente, pero con el modelo de datos del que disponemos no se pueden correlacionar ambas columnas.
- Al hacer el cruce del campo CUSTOMER_ID con el campo ID_CLIENTE de la tabla STAGE.STG_CLIENTES_CRM vemos que hay 28 registros que corresponden a CUSTOMER_ID que no aparecen en la tabla STG_CLIENTES_CRM:

```sql
SELECT PRODUCT_ID ID_SERVICIO
, CUSTOMER_ID, CLI.ID_CLIENTE
, ACCESS_POINT PUNTO_ACCESO
, CHANNEL
, AGENT_CODE ID_AGENTE
, START_DATE FC_INICIO
, INSTALL_DATE FC_INSTALACION
, END_DATE FC_FIN
, PRODUCT_CITY
, PRODUCT_ADDRESS
, PRODUCT_POSTAL_CODE
, PRODUCT_STATE
, PRODUCT_COUNTRY
FROM STAGE.STG_PRODUCTOS_CRM SER
LEFT JOIN ODS.ODS_HC_CLIENTES CLI ON CLI.ID_CLIENTE = SER.CUSTOMER_ID
WHERE ID_CLIENTE IS NULL
```

  Si ordenamos los resultados, en total nos salen tres CUSTOMER_ID no registrados en la tabla de clientes: el 10000, 14826 y 16689. Yo lo que haría en este caso es reunirme con la gente del operacional, para que me aclaren si los clientes que se dan de baja se borran de la tabla CLIENTES del CRM o de si se trata de un error que habría que intentar corregir, ya que tenemos tres clientes contratando servicios y de los que no tenemos información.
- La tabla se llama PRODUCTOS, entendiéndose como servicios que contrata el cliente, descritos por el campo PRODUCT_NAME, que tiene un total de 6 valores distintos. Por lo tanto, sacaremos los productos a una dimensión y la tabla PRODUCTOS del CRM pasará a ser nuestra tabla SERVICIOS.
- El campo PRODUCT_ID será la clave primaria de nuestra tabla servicios, con los resultados que nos ha dado vemos que siempre está relleno y que los valores no se repiten, por lo que puede hacer de clave primaria.
- Otra dimensión que es interesante tener es el canal, y por supuesto tenemos la dirección que tirará de las tablas de direcciones que creamos previamente para los

clientes. Para ello deberemos insertar las nuevas direcciones que no aparezcan en la tabla de clientes.

1. Creamos las tablas del modelo de servicios:

Creacion modelo servicios.sql

```
USE ODS;

DROP TABLE IF EXISTS ODS_DM_CANALES;

CREATE TABLE ODS_DM_CANALES
(ID_CANAL INT unsigned auto_increment PRIMARY KEY
, DE_CANAL VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS_DM_PRODUCTOS;

CREATE TABLE ODS_DM_PRODUCTOS
(ID_PRODUCTO INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_PRODUCTO VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);

DROP TABLE IF EXISTS ODS_HC_SERVICIOS;

CREATE TABLE ODS_HC_SERVICIOS
(ID_SERVICIO INT NOT NULL PRIMARY KEY
, ID_CLIENTE INT(11)
, ID_PRODUCTO INT(10)
, PUNTO_ACCESO VARCHAR(512)
, ID_CANAL INT(10)
, ID_AGENTE INT(11)
, ID_DIRECCION_SERVICIO INT(10)
, FC_INICIO DATETIME
, FC_INSTALACION DATETIME
, FC_FIN DATETIME
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME);
```

2. Creamos las FK del modelo de servicios:

FK modelo servicios.sql

USE ODS;

```
/*
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX FK_SER_CLI_IDX (ID_CLIENTE
ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT FK_SER_CLI FOREIGN KEY
(ID_CLIENTE)
  REFERENCES ODS.ODS_HC_CLIENTES (ID_CLIENTE);
*/

ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX FK_SER_PRO_IDX (ID_PRODUCTO
ASC);

ALTER TABLE ODS.ODS_HC_SERVICIOS CHANGE COLUMN ID_PRODUCTO ID_PRODUCTO
INT(10) UNSIGNED;
```

```
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT FK_SER_PRO FOREIGN KEY
(ID_PRODUCTO)
  REFERENCES ODS.ODS_DM_PRODUCTOS (ID_PRODUCTO);

ALTER TABLE ODS.ODS_HC_SERVICIOS CHANGE COLUMN ID_CANAL ID_CANAL INT(10)
UNSIGNED;
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX FK_SER_CANAL_IDX (ID_CANAL
ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT FK_SER_CANAL FOREIGN KEY
(ID_CANAL)
  REFERENCES ODS.ODS_DM_CANALES (ID_CANAL);

ALTER TABLE ODS.ODS_HC_SERVICIOS CHANGE COLUMN ID_DIRECCION_SERVICIO
ID_DIRECCION_SERVICIO INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD INDEX FK_SER_DIR_IDX
(ID_DIRECCION_SERVICIO ASC);
ALTER TABLE ODS.ODS_HC_SERVICIOS ADD CONSTRAINT FK_SER_DIR FOREIGN KEY
(ID_DIRECCION_SERVICIO)
  REFERENCES ODS.ODS_HC_DIRECCIONES (ID_DIRECCION);
```

3. Poblamos el modelo de servicios:

POBLAR_ODS_SERVICIOS.sql

```
USE ODS;

INSERT INTO ODS_DM_CANALES (DE_CANAL, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(PRO.CHANNEL)), NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM PRO
WHERE TRIM(PRO.CHANNEL)<>'';

INSERT INTO ODS_DM_CANALES VALUES (99, 'DESCONOCIDO', NOW(),NOW());
INSERT INTO ODS_DM_CANALES VALUES (98, 'NO APLICA', NOW(),NOW());

COMMIT;

ANALYZE TABLE ODS_DM_CANALES;


INSERT INTO ODS.ODS_DM_PRODUCTOS (DE_PRODUCTO, FC_INSERT,
FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(PRO.PRODUCT_NAME)), NOW(), NOW()
FROM STAGE.STG_PRODUCTOS_CRM PRO
WHERE TRIM(PRO.PRODUCT_NAME)<>'';

INSERT INTO ODS_DM_PRODUCTOS VALUES (99, 'DESCONOCIDO', NOW(),NOW());
INSERT INTO ODS_DM_PRODUCTOS VALUES (98, 'NO APLICA', NOW(),NOW());

COMMIT;

ANALYZE TABLE ODS_DM_PRODUCTOS;
```

POBLAR_ODS_SERVICIOS II.sql

```
USE ODS;

DROP TABLE IF EXISTS TMP_DIRECCIONES_CLIENTES;

CREATE TABLE TMP_DIRECCIONES_CLIENTES AS
SELECT DIR.ID_DIRECCION
, DIR.DE_DIRECCION
, DIR.DE_CP
```

```
, CIU.DE_CIUDAD
, CIU.DE_ESTADO
, PAI.DE_PAIS
FROM ODS.ODS_HC_DIRECCIONES DIR
INNER JOIN ODS.ODS_DM_CIUDADES_ESTADO CIU ON
DIR.ID_CIUDAD_ESTADO=CIU.ID_CIUDAD_ESTADO
INNER JOIN ODS.ODS_DM_PAISES PAI ON CIU.ID_PAIS=PAI.ID_PAIS;

ANALYZE TABLE TMP_DIRECCIONES_CLIENTES;

DROP TABLE IF EXISTS TMP_DIR_SERVICIO;

CREATE TABLE TMP_DIR_SERVICIO AS
SELECT DISTINCT CASE WHEN TRIM(PRODUCT_ADDRESS)<>'' THEN
UPPER(TRIM(PRODUCT_ADDRESS)) ELSE 'DESCONOCIDO' END PRODUCT_ADDRESS
, CASE WHEN TRIM(PRODUCT_POSTAL_CODE)<>'' THEN
UPPER(TRIM(PRODUCT_POSTAL_CODE)) ELSE 99999 END PRODUCT_POSTAL_CODE
, CASE WHEN TRIM(PRODUCT_CITY)<>'' THEN UPPER(TRIM(PRODUCT_CITY)) ELSE
'DESCONOCIDO' END PRODUCT_CITY
, CASE WHEN TRIM(PRODUCT_STATE)<>'' THEN UPPER(TRIM(PRODUCT_STATE)) ELSE
'DESCONOCIDO' END PRODUCT_STATE
, CASE WHEN TRIM(PRODUCT_COUNTRY)<>'' THEN
UPPER(TRIM(REPLACE(PROD.PRODUCT_COUNTRY,'United States','US'))) ELSE
'DESCONOCIDO' END  PRODUCT_COUNTRY
FROM STAGE.STG_PRODUCTOS_CRM PROD
WHERE TRIM(PRODUCT_ADDRESS)<>'' OR TRIM(PRODUCT_POSTAL_CODE)<>'' OR
TRIM(PRODUCT_CITY)<>'' OR TRIM(PRODUCT_STATE)<>'' OR
TRIM(PRODUCT_COUNTRY)<>''
;

DROP TABLE IF EXISTS TMP_DIR_SERVICIO2;

CREATE TABLE TMP_DIR_SERVICIO2 AS
SELECT PRODUCT_ADDRESS, PRODUCT_POSTAL_CODE, PRODUCT_CITY,
PRODUCT_STATE, PRODUCT_COUNTRY
FROM TMP_DIR_SERVICIO SERV
LEFT OUTER JOIN TMP_DIRECCIONES_CLIENTES CLI ON CONCAT(PRODUCT_ADDRESS,
PRODUCT_POSTAL_CODE, PRODUCT_CITY, PRODUCT_STATE,
PRODUCT_COUNTRY)=CONCAT(DE_DIRECCION, DE_CP, DE_CIUDAD, DE_ESTADO,
DE_PAIS)
WHERE CONCAT(DE_DIRECCION, DE_CP, DE_CIUDAD, DE_ESTADO, DE_PAIS) IS
NULL;

INSERT INTO ODS_DM_PAISES (DE_PAIS, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT PAI.DE_PAIS, NOW(), NOW()
FROM TMP_DIR_SERVICIO2 DIR
INNER JOIN ODS.ODS_DM_PAISES PAI ON DIR.PRODUCT_COUNTRY=PAI.DE_PAIS
LEFT OUTER JOIN TMP_DIRECCIONES_CLIENTES CLI ON
PRODUCT_COUNTRY=CLI.DE_PAIS
WHERE CLI.DE_PAIS IS NULL;

COMMIT;

INSERT INTO ODS_DM_CIUDADES_ESTADO (DE_CIUDAD, DE_ESTADO, ID_PAIS,
FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT PRODUCT_CITY, PRODUCT_STATE, PAI.ID_PAIS, NOW(), NOW()
FROM TMP_DIR_SERVICIO2 DIR
INNER JOIN ODS.ODS_DM_PAISES PAI ON DIR.PRODUCT_COUNTRY=PAI.DE_PAIS
LEFT OUTER JOIN TMP_DIRECCIONES_CLIENTES CLI ON CONCAT(PRODUCT_CITY,
PRODUCT_STATE, PRODUCT_COUNTRY)=CONCAT(CLI.DE_CIUDAD, CLI.DE_ESTADO,
CLI.DE_PAIS)
WHERE CONCAT(CLI.DE_CIUDAD, CLI.DE_ESTADO, CLI.DE_PAIS) IS NULL;
```

```
COMMIT;

UPDATE ODS_DM_CIUDADES_ESTADO SET ID_CIUDAD_ESTADO=ID_CIUDAD_ESTADO-917
WHERE ID_CIUDAD_ESTADO>999;

COMMIT;


INSERT INTO ODS_HC_DIRECCIONES (DE_DIRECCION, DE_CP, ID_CIUDAD_ESTADO,
FC_INSERT, FC_MODIFICACION)
SELECT PRODUCT_ADDRESS, PRODUCT_POSTAL_CODE, CIU.ID_CIUDAD_ESTADO,
NOW(), NOW()
FROM TMP_DIR_SERVICIO2 DIR
INNER JOIN ODS.ODS_DM_PAISES PAI ON DIR.PRODUCT_COUNTRY=PAI.DE_PAIS
INNER JOIN ODS.ODS_DM_CIUDADES_ESTADO CIU ON CONCAT(DIR.PRODUCT_CITY,
PRODUCT_STATE)=CONCAT(CIU.DE_CIUDAD, CIU.DE_ESTADO);

COMMIT;

UPDATE ODS_HC_DIRECCIONES SET ID_DIRECCION=ID_DIRECCION-982502 WHERE
ID_DIRECCION>999999;

COMMIT;

DROP TABLE IF EXISTS TMP_DIRECCIONES_CLIENTES;
DROP TABLE IF EXISTS TMP_DIR_SERVICIO;
DROP TABLE IF EXISTS TMP_DIR_SERVICIO2;
```

POBLAR_ODS_SERVICIOS III.sql:

```
USE ODS;

DROP TABLE IF EXISTS TMP_DIR_SERVICIO;

CREATE TABLE TMP_DIR_SERVICIO AS
SELECT DIR.ID_DIRECCION,
DIR.DE_DIRECCION,
DIR.DE_CP,
CIU.DE_CIUDAD,
CIU.DE_ESTADO,
PAI.DE_PAIS
FROM ODS.ODS_HC_DIRECCIONES DIR
INNER JOIN ODS.ODS_DM_CIUDADES_ESTADO CIU ON DIR.ID_CIUDAD_ESTADO =
CIU.ID_CIUDAD_ESTADO
INNER JOIN ODS.ODS_DM_PAISES PAI ON CIU.ID_PAIS = PAI.ID_PAIS;

ANALYZE TABLE TMP_DIR_SERVICIO;


DROP TABLE IF EXISTS TMP_DIR_SERVICIO2;

CREATE TABLE TMP_DIR_SERVICIO2 AS
SELECT PRO.PRODUCT_ID
, DIR.ID_DIRECCION
FROM STAGE.STG_PRODUCTOS_CRM PRO
INNER JOIN ODS.TMP_DIR_SERVICIO DIR ON CASE WHEN
TRIM(PRO.PRODUCT_ADDRESS)<>'' THEN UPPER(TRIM(PRO.PRODUCT_ADDRESS)) ELSE
'DESCONOCIDO' END = DIR.DE_DIRECCION
AND CASE WHEN TRIM(PRO.PRODUCT_POSTAL_CODE)<>'' THEN
TRIM(PRO.PRODUCT_POSTAL_CODE) ELSE 99999 END = DIR.DE_CP
AND CASE WHEN TRIM(PRO.PRODUCT_STATE)<>'' THEN
UPPER(TRIM(PRO.PRODUCT_STATE)) ELSE 'DESCONOCIDO' END = DIR.DE_ESTADO
```

```
AND CASE WHEN TRIM(PRO.PRODUCT_CITY)<>'' THEN
UPPER(TRIM(PRO.PRODUCT_CITY)) ELSE 'DESCONOCIDO' END = DIR.DE_CIUDAD
AND CASE WHEN TRIM(PRO.PRODUCT_COUNTRY)<>'' THEN
UPPER(TRIM(REPLACE(PRO.PRODUCT_COUNTRY,'United States','US'))) ELSE
'DESCONOCIDO' END = DIR.DE_PAIS;

ANALYZE TABLE TMP_DIR_SERVICIO2;


INSERT INTO ODS.ODS_HC_SERVICIOS
SELECT SER.PRODUCT_ID ID_SERVICIO
, CASE WHEN CLI.ID_CLIENTE IS NULL THEN CAST(LPAD(CUSTOMER_ID,9,9) AS
SIGNED INTEGER) ELSE ID_CLIENTE END ID_CLIENTE
, PRO.ID_PRODUCTO
, ACCESS_POINT PUNTO_ACCESO
, CA.ID_CANAL
, CASE WHEN TRIM(SER.AGENT_CODE)='' THEN 9998 ELSE SER.AGENT_CODE END
ID_AGENTE
, ID_DIRECCION
, CASE WHEN TRIM(SER.START_DATE) = '' THEN
STR_TO_DATE('31/12/9999','%d/%m/%Y') ELSE
STR_TO_DATE(SER.START_DATE,'%d/%m/%Y') END FC_INICIO
, CASE WHEN TRIM(SER.INSTALL_DATE) = '' THEN
STR_TO_DATE('31/12/9998','%d/%m/%Y') ELSE
STR_TO_DATE(SUBSTRING(INSTALL_DATE, 1,19),'%Y-%m-%d %H:%i:%s') END
FC_INSTALACION
, CASE WHEN TRIM(SER.END_DATE) = '' THEN
STR_TO_DATE('31/12/9998','%d/%m/%Y') ELSE
STR_TO_DATE(SUBSTRING(END_DATE, 1,19),'%Y-%m-%d %H:%i:%s') END FC_FIN
, NOW()
, STR_TO_DATE('31/12/9998','%d/%m/%Y')
FROM STAGE.STG_PRODUCTOS_CRM SER
LEFT JOIN ODS.ODS_HC_CLIENTES CLI ON CLI.ID_CLIENTE = SER.CUSTOMER_ID
INNER JOIN ODS.ODS_DM_PRODUCTOS PRO ON CASE WHEN
TRIM(SER.PRODUCT_NAME)<>'' THEN UPPER(TRIM(SER.PRODUCT_NAME)) ELSE
'DESCONOCIDO' END = PRO.DE_PRODUCTO
INNER JOIN ODS.ODS_DM_CANALES CA ON CASE WHEN TRIM(SER.CHANNEL)<>'' THEN
UPPER(TRIM(SER.CHANNEL)) ELSE 'DESCONOCIDO' END = CA.DE_CANAL
INNER JOIN ODS.TMP_DIR_SERVICIO2 DIR ON DIR.PRODUCT_ID = SER.PRODUCT_ID;

COMMIT;

ANALYZE TABLE ODS.ODS_HC_SERVICIOS;

DROP TABLE ODS.TMP_DIR_SERVICIO;
DROP TABLE ODS.TMP_DIR_SERVICIO2;
```

b) Estudio de la tabla STG_FACTURAS_FCT:

Analisis STG_FACTURAS_FCT.sql

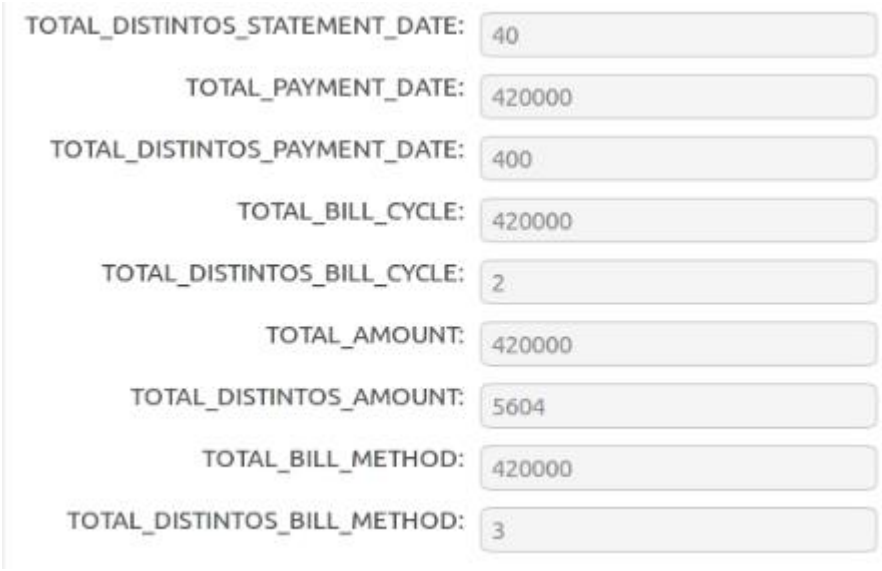
```
USE STAGE;

SELECT COUNT(*) TOTAL_REGISTROS
, SUM(CASE WHEN LENGTH(TRIM(BILL_REF_NO))<>0 THEN 1 ELSE 0 END)
TOTAL_BILL_REF_NO
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL_REF_NO))<>0 THEN BILL_REF_NO
ELSE 0 END) TOTAL_DISTINTOS_BILL_REF_NO
, SUM(CASE WHEN LENGTH(TRIM(CUSTOMER_ID))<>0 THEN 1 ELSE 0 END)
TOTAL_CUSTOMER_ID
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(CUSTOMER_ID))<>0 THEN CUSTOMER_ID
ELSE 0 END) TOTAL_DISTINTOS_CUSTOMER_ID
```

```
, SUM(CASE WHEN LENGTH(TRIM(START_DATE))<>0 THEN 1 ELSE 0 END)
TOTAL_START_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(START_DATE))<>0 THEN START_DATE
ELSE 0 END) TOTAL_DISTINTOS_START_DATE
, SUM(CASE WHEN LENGTH(TRIM(END_DATE))<>0 THEN 1 ELSE 0 END)
TOTAL_END_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(END_DATE))<>0 THEN END_DATE ELSE
0 END) TOTAL_DISTINTOS_END_DATE
, SUM(CASE WHEN LENGTH(TRIM(STATEMENT_DATE))<>0 THEN 1 ELSE 0 END)
TOTAL_STATEMENT_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(STATEMENT_DATE))<>0 THEN
STATEMENT_DATE ELSE 0 END) TOTAL_DISTINTOS_STATEMENT_DATE
, SUM(CASE WHEN LENGTH(TRIM(PAYMENT_DATE))<>0 THEN 1 ELSE 0 END)
TOTAL_PAYMENT_DATE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PAYMENT_DATE))<>0 THEN
PAYMENT_DATE ELSE 0 END) TOTAL_DISTINTOS_PAYMENT_DATE
, SUM(CASE WHEN LENGTH(TRIM(BILL_CYCLE))<>0 THEN 1 ELSE 0 END)
TOTAL_BILL_CYCLE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL_CYCLE))<>0 THEN BILL_CYCLE
ELSE 0 END) TOTAL_DISTINTOS_BILL_CYCLE
, SUM(CASE WHEN LENGTH(TRIM(AMOUNT))<>0 THEN 1 ELSE 0 END) TOTAL_AMOUNT
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AMOUNT))<>0 THEN AMOUNT ELSE 0
END) TOTAL_DISTINTOS_AMOUNT
, SUM(CASE WHEN LENGTH(TRIM(BILL_METHOD))<>0 THEN 1 ELSE 0 END)
TOTAL_BILL_METHOD
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(BILL_METHOD))<>0 THEN BILL_METHOD
ELSE 0 END) TOTAL_DISTINTOS_BILL_METHOD
FROM STAGE.STG_FACTURAS_FCT;
```

Esta consulta nos da como resultado:

TOTAL_REGISTROS:	420000
TOTAL_BILL_REF_NO:	420000
TOTAL_DISTINTOS_BILL_REF_NO:	420000
TOTAL_CUSTOMER_ID:	420000
TOTAL_DISTINTOS_CUSTOMER_ID:	20000
TOTAL_START_DATE:	420000
TOTAL_DISTINTOS_START_DATE:	40
TOTAL_END_DATE:	420000
TOTAL_DISTINTOS_END_DATE:	20
TOTAL_STATEMENT_DATE:	420000



Conclusiones:

- Esta tabla está rellena por completo, no existiendo nulos en ninguno de los campos.
- Observamos que existen tres métodos de pago y dos ciclos de facturación, por lo que deberemos sacar estos campos que se repiten continuamente a dimensiones.
- También observamos que hay 20000 clientes con facturas, pero en nuestra tabla de clientes sólo están registrados 17558. Haciendo la correspondiente select para obtener los clientes que aparecen en las facturas y que no están en la tabla de clientes:

```
SELECT DISTINCT(FA.CUSTOMER_ID) FROM STAGE.STG_FACTURAS_FCT FA
LEFT OUTER JOIN ODS.ODS_HC_CLIENTES CLI ON FA.CUSTOMER_ID =
CLI.ID_CLIENTE WHERE CLI.ID_CLIENTE IS NULL;
```

Obtenemos los ids de los 2442 clientes sin registrar y a los cuales les estamos facturando. Dos de ellos (14826 y 16689) ya nos aparecían como clientes que tenían contratado un servicio y que no estaban registrados al hacer el análisis de la tabla de productos. Esto habría que comentarlo con las personas que se encargan del operacional para corregirlo, porque es un 12% de clientes de los cuales no tenemos información en la tabla de clientes y les estamos facturando.

1. Creamos las tablas del modelo facturas:

Creacion modelo facturas.sql

```
USE ODS;

DROP TABLE IF EXISTS ODS_DM_METODOS_PAGO;

CREATE TABLE ODS_DM_METODOS_PAGO
(ID_METODO_PAGO INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_METODO_PAGO VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
```

```
);

DROP TABLE IF EXISTS ODS_DM_CICLOS_FACTURACION;

CREATE TABLE ODS_DM_CICLOS_FACTURACION
(ID_CICLO_FACTURACION INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_CICLO_FACTURACION VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);

DROP TABLE IF EXISTS ODS_HC_FACTURAS;

CREATE TABLE ODS_HC_FACTURAS
(
ID_FACTURA INT UNSIGNED PRIMARY KEY
, ID_CLIENTE INT(11)
, FC_INICIO DATETIME
, FC_FIN DATETIME
, FC_ESTADO DATETIME
, FC_PAGO DATETIME
, ID_CICLO_FACTURACION INT(10)
, ID_METODO_PAGO INT(10)
, ID_MONEDA INT(10)
, CANTIDAD INT(11)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);
```

2. Creamos las Foreing Keys:

FK modelo facturas.sql

```
USE ODS;
/*
ALTER TABLE ODS.ODS_HC_FACTURAS ADD INDEX FK_FAC_CLI_IDX (ID_CLIENTE
ASC);
ALTER TABLE ODS.ODS_HC_FACTURAS ADD CONSTRAINT FK_FAC_CLI FOREIGN KEY
(ID_CLIENTE)
  REFERENCES ODS.ODS_HC_CLIENTES (ID_CLIENTE);
*/
ALTER TABLE ODS.ODS_HC_FACTURAS CHANGE COLUMN ID_METODO_PAGO
ID_METODO_PAGO INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_FACTURAS ADD INDEX FK_FAC_MET_IDX (ID_METODO_PAGO
ASC);
ALTER TABLE ODS.ODS_HC_FACTURAS ADD CONSTRAINT FK_FAC_MET FOREIGN KEY
(ID_METODO_PAGO)
  REFERENCES ODS.ODS_DM_METODOS_PAGO (ID_METODO_PAGO);

ALTER TABLE ODS.ODS_HC_FACTURAS CHANGE COLUMN ID_CICLO_FACTURACION
ID_CICLO_FACTURACION INT(10) UNSIGNED;
ALTER TABLE ODS.ODS_HC_FACTURAS ADD INDEX FK_FAC_CIC_IDX
(ID_CICLO_FACTURACION ASC);
ALTER TABLE ODS.ODS_HC_FACTURAS ADD CONSTRAINT FK_FAC_CIC FOREIGN KEY
(ID_CICLO_FACTURACION)
  REFERENCES ODS.ODS_DM_CICLOS_FACTURACION (ID_CICLO_FACTURACION);
```

3. Poblamos el modelo facturas:

POBLAR_ODS_FACTURAS.sql

```
USE ODS;

INSERT INTO ODS.ODS_DM_METODOS_PAGO VALUES (1, 'DIRECT DEBIT', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_METODOS_PAGO VALUES (2, 'CREDIT CARD', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_METODOS_PAGO VALUES (3, 'CHECK PAYMENT', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_METODOS_PAGO VALUES (99, 'DESCONOCIDO', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_METODOS_PAGO VALUES (98, 'NO APLICA', NOW(),
NOW());

COMMIT;

INSERT INTO ODS.ODS_DM_CICLOS_FACTURACION VALUES (1, 'M01', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_CICLOS_FACTURACION VALUES (2, 'M15', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_CICLOS_FACTURACION VALUES (99, 'DESCONOCIDO',
NOW(), NOW());
INSERT INTO ODS.ODS_DM_CICLOS_FACTURACION VALUES (98, 'NO APLICA',
NOW(), NOW());

COMMIT;


INSERT INTO ODS.ODS_HC_FACTURAS
SELECT CAST(FAC.BILL_REF_NO AS UNSIGNED) ID_FACTURA
, CASE WHEN CLI.ID_CLIENTE IS NULL THEN CAST(LPAD(FAC.CUSTOMER_ID,9,9)
AS SIGNED INTEGER) ELSE CLI.ID_CLIENTE END ID_CLIENTE
, CASE WHEN TRIM(FAC.START_DATE)<>'' THEN
STR_TO_DATE(TRIM(FAC.START_DATE),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s') END FC_INICIO
, CASE WHEN TRIM(FAC.END_DATE)<>'' THEN
STR_TO_DATE(TRIM(FAC.END_DATE),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s') END FC_FIN
, CASE WHEN TRIM(FAC.STATEMENT_DATE)<>'' THEN
STR_TO_DATE(TRIM(FAC.STATEMENT_DATE),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s') END FC_ESTADO
, CASE WHEN TRIM(FAC.PAYMENT_DATE)<>'' THEN
STR_TO_DATE(TRIM(FAC.PAYMENT_DATE),'%Y-%m-%d %H:%i:%s') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s') END FC_PAGO
, CI.ID_CICLO_FACTURACION
, ME.ID_METODO_PAGO
, CASE WHEN TRIM(FAC.AMOUNT)<>'' THEN CAST(AMOUNT AS DECIMAL(9,2)) ELSE
'9999999.99' END CANTIDAD
, NOW() FC_INSERT
, STR_TO_DATE('9998-12-31 00:00:00','%Y-%m-%d %H:%i:%s') FC_MODIFICA
FROM STAGE.STG_FACTURAS_FCT FAC
LEFT OUTER JOIN ODS.ODS_HC_CLIENTES CLI ON CASE WHEN
TRIM(CUSTOMER_ID)<>'' THEN TRIM(FAC.CUSTOMER_ID) ELSE 'DESCONOCIDO' END
= CLI.ID_CLIENTE
INNER JOIN ODS.ODS_DM_CICLOS_FACTURACION CI ON CASE WHEN
TRIM(FAC.BILL_CYCLE)<>'' THEN UPPER(TRIM(BILL_CYCLE)) ELSE 'DESCONOCIDO'
END = CI.DE_CICLO_FACTURACION
INNER JOIN ODS.ODS_DM_METODOS_PAGO ME ON CASE WHEN
TRIM(FAC.BILL_METHOD)<>'' THEN UPPER(TRIM(BILL_METHOD)) ELSE
'DESCONOCIDO' END = ME.DE_METODO_PAGO;INNER JOIN
```

```
ODS.ODS_DM_CICLOS_FACTURACION CI ON CASE WHEN TRIM(FAC.BILL_CYCLE)<>''
THEN UPPER(TRIM(BILL_CYCLE)) ELSE 'DESCONOCIDO' END =
CI.DE_CICLO_FACTURACION
INNER JOIN ODS.ODS_DM_METODOS_PAGO ME ON CASE WHEN
TRIM(FAC.BILL_METHOD)<>'' THEN UPPER(TRIM(BILL_METHOD)) ELSE
'DESCONOCIDO' END = ME.DE_METODO_PAGO;

COMMIT;
```

c) Estudio de STG_CONTACTOS_IVR:

Analisis STG_CONTACTOS_IVR.sql

```
USE STAGE;

SELECT COUNT(*) TOTAL_REGISTROS
, SUM(CASE WHEN LENGTH(TRIM(ID))<>0 THEN 1 ELSE 0 END) TOTAL_ID
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(ID))<>0 THEN ID ELSE 0 END)
TOTAL_DISTINTOS_ID
, SUM(CASE WHEN LENGTH(TRIM(PHONE_NUMBER))<>0 THEN 1 ELSE 0 END)
TOTAL_PHONE_NUMBER
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PHONE_NUMBER))<>0 THEN
PHONE_NUMBER ELSE 0 END) TOTAL_DISTINTOS_PHONE_NUMBER
, SUM(CASE WHEN LENGTH(TRIM(START_DATETIME))<>0 THEN 1 ELSE 0 END)
TOTAL_START_DATETIME
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(START_DATETIME))<>0 THEN
START_DATETIME ELSE 0 END) TOTAL_DISTINTOS_START_DATETIME
, SUM(CASE WHEN LENGTH(TRIM(END_DATETIME))<>0 THEN 1 ELSE 0 END)
TOTAL_END_DATETIME
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(END_DATETIME))<>0 THEN
END_DATETIME ELSE 0 END) TOTAL_DISTINTOS_END_DATETIME
, SUM(CASE WHEN LENGTH(TRIM(SERVICE))<>0 THEN 1 ELSE 0 END)
TOTAL_SERVICE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(SERVICE))<>0 THEN SERVICE ELSE 0
END) TOTAL_DISTINTOS_SERVICE
, SUM(CASE WHEN LENGTH(TRIM(FLG_TRANSFER))<>0 THEN 1 ELSE 0 END)
TOTAL_FLG_TRANSFER
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(FLG_TRANSFER))<>0 THEN
FLG_TRANSFER ELSE 0 END) TOTAL_DISTINTOS_FLG_TRANSFER
, SUM(CASE WHEN LENGTH(TRIM(AGENT))<>0 THEN 1 ELSE 0 END) TOTAL_AGENT
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AGENT))<>0 THEN AGENT ELSE 0 END)
TOTAL_DISTINTOS_AGENT
FROM STAGE.STG_CONTACTOS_IVR;
```

Esta consulta nos da como resultado:

TOTAL_REGISTROS:	202717
TOTAL_ID:	202717
TOTAL_DISTINTOS_ID:	150000
TOTAL_PHONE_NUMBER:	185018
TOTAL_DISTINTOS_PHONE_NUMBER:	18226
TOTAL_START_DATETIME:	202717
TOTAL_DISTINTOS_START_DATETIME:	201098
TOTAL_END_DATETIME:	186535
TOTAL_DISTINTOS_END_DATETIME:	183678
TOTAL_SERVICE:	202502
TOTAL_DISTINTOS_SERVICE:	7
TOTAL_FLG_TRANSFER:	202717
TOTAL_DISTINTOS_FLG_TRANSFER:	2
TOTAL_AGENT:	194739
TOTAL_DISTINTOS_AGENT:	594

Conclusiones:

- Vemos que el ID se repite, por lo que no puede ser PRIMARY KEY de nuestra tabla de llamadas. Pondremos entonces un ID autogenerado para que sea la clave primaria y después un campo que guarde el valor del ID_IVR.
- Para un mismo ID_IVR aparecen distintos números de teléfono, además si los ordenamos por START_DATETIME, coincide la fecha END_DATETIME del primer registro con el valor de START_DATETIME del siguiente registro y así sucesivamente hasta que el último registro tiene FLG_TRANSFER a false… podemos deducir entonces que los distintos teléfonos se corresponden a la transferencia de llamadas entre distintos agentes y/o departamentos, no siendo necesariamente los teléfonos de los clientes.
- Intentamos establecer una relación entre los clientes guardados en ODS_HC_CLIENTES y los que aparecen en la tabla de llamadas (cruzando por el campo del número de teléfono):

```
USE ODS;

CREATE TABLE TMP_ID_CLIE_TELEFONO
SELECT MAX(ID_CLIENTE),
TELEFONO_CLIENTE
FROM ODS.ODS_HC_CLIENTES
GROUP BY TELEFONO_CLIENTE;

SELECT * FROM TMP_ID_CLIE_TELEFONO CLI
INNER JOIN STAGE.STG_CONTACTOS_IVR CON ON CLI.TELEFONO_CLIENTE =
CAST(CON.PHONE_NUMBER AS UNSIGNED INTEGER);
```

Este cruce no me devuelve resultados, por lo que no disponemos de información suficiente como para establecer esta relación.

- En el campo de AGENT encontramos un texto que podría corresponder con el nombre del usuario del agente. No podemos relacionar este campo con el AGENT_CODE que aparecía en STAGE.STG_PRODUCTOS_CRM, así que en principio la dimensión de agentes estará sólo relacionada con la tabla de LLAMADAS y en el caso de la tabla SERVICIOS el campo ID_AGENTE quedará sin relacionar.

1. Creamos las tablas del modelo de llamadas:

Creacion modelo llamadas.sql

```
USE ODS;

DROP TABLE IF EXISTS ODS_DM_DEPARTAMENTOS_CC;

CREATE TABLE ODS_DM_DEPARTAMENTOS_CC
(ID_DEPARTAMENTO_CC INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_DEPARTAMENTO_CC VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_AGENTES_CC;

CREATE TABLE ODS_DM_AGENTES_CC
(ID_AGENTE_CC INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_AGENTE_CC VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);

DROP TABLE IF EXISTS ODS_HC_LLAMADAS;

CREATE TABLE ODS_HC_LLAMADAS
(
ID_LLAMADA INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, ID_IVR INT(11)
, TELEFONO_LLAMADA BIGINT(20)
, ID_CLIENTE INT(11)
, FC_INICIO_LLAMADA DATETIME
, FC_FIN_LLAMADA DATETIME
, ID_DEPARTAMENTO_CC INT(10) UNSIGNED
, FLG_TRANSFERIDO TINYINT(1)
, ID_AGENTE_CC INT(10) UNSIGNED
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
```

```
);
```

2. Creamos las Foreing Keys:

FK modelo llamadas.sql

```
USE ODS;

ALTER TABLE ODS.ODS_HC_LLAMADAS ADD INDEX FK_LLA_CLI_IDX (ID_CLIENTE
ASC);
ALTER TABLE ODS.ODS_HC_LLAMADAS ADD CONSTRAINT FK_LLA_CLI FOREIGN KEY
(ID_CLIENTE)
  REFERENCES ODS.ODS_HC_CLIENTES (ID_CLIENTE);

ALTER TABLE ODS.ODS_HC_LLAMADAS ADD INDEX FK_LLA_DEP_IDX
(ID_DEPARTAMENTO_CC ASC);
ALTER TABLE ODS.ODS_HC_LLAMADAS ADD CONSTRAINT FK_LLA_DEP FOREIGN KEY
(ID_DEPARTAMENTO_CC)
  REFERENCES ODS.ODS_DM_DEPARTAMENTOS_CC (ID_DEPARTAMENTO_CC);

ALTER TABLE ODS.ODS_HC_LLAMADAS ADD INDEX FK_LLA_AGEN_IDX (ID_AGENTE_CC
ASC);
ALTER TABLE ODS.ODS_HC_LLAMADAS ADD CONSTRAINT FK_LLA_AGEN FOREIGN KEY
(ID_AGENTE_CC)
  REFERENCES ODS.ODS_DM_AGENTES_CC (ID_AGENTE_CC);
```

3. Poblamos el modelo llamadas:

POBLAR_ODS_LLAMADAS.sql

```
USE ODS;

INSERT INTO ODS.ODS_DM_DEPARTAMENTOS_CC (DE_DEPARTAMENTO_CC, FC_INSERT,
FC_MODIFICACION)
SELECT DISTINCT CASE WHEN TRIM(SERVICE)<>'' THEN UPPER(TRIM(SERVICE))
ELSE 'DESCONOCIDO' END DE_DEPARTAMENTO_CC
, NOW(), NOW()
FROM STAGE.STG_CONTACTOS_IVR;

COMMIT;

INSERT INTO ODS.ODS_DM_AGENTES_CC (DE_AGENTE_CC, FC_INSERT,
FC_MODIFICACION)
SELECT DISTINCT CASE WHEN TRIM(AGENT)<>'' THEN UPPER(TRIM(AGENT)) ELSE
'DESCONOCIDO' END DE_AGENTE_CC,
NOW(), NOW()
FROM STAGE.STG_CONTACTOS_IVR;

COMMIT;

/*
Tenemos que insertar en clientes el registro para el cliente
'DESCONOCIDO', para que no nos de este error:
Error Code: 1452. Cannot add or update a child row: a foreign key
constraint fails (`ODS`.`ODS_HC_LLAMADAS`, CONSTRAINT `FK_LLA_CLI`
FOREIGN KEY (`ID_CLIENTE`) REFERENCES `ODS_HC_CLIENTES` (`ID_CLIENTE`))
*/
INSERT INTO ODS.ODS_HC_CLIENTES VALUES (999999999, 'DESCONOCIDO',
'DESCONOCIDO','99-999-9999', 99, 999999,9999999999,
'DESCONOCIDO@DESCONOCIDO.ES',STR_TO_DATE('31/12/9999','%d/%m/%Y'),999,99
9,NOW(),STR_TO_DATE('31/12/9999','%d/%m/%Y'));
```

```
COMMIT;

INSERT INTO ODS.ODS_HC_LLAMADAS (ID_IVR, TELEFONO_LLAMADA, ID_CLIENTE,
FC_INICIO_LLAMADA, FC_FIN_LLAMADA, ID_DEPARTAMENTO_CC,
FLG_TRANSFERIDO,ID_AGENTE_CC, FC_INSERT, FC_MODIFICACION)
SELECT CAST(ID AS SIGNED INTEGER) ID_IVR
, CASE WHEN TRIM(LLA.PHONE_NUMBER)<>'' THEN CAST(TRIM(LLA.PHONE_NUMBER)
AS SIGNED INTEGER) ELSE 9999999999 END TELEFONO_LLAMADA
, 999999999 ID_CLIENTE
, CASE WHEN TRIM(LLA.START_DATETIME)<>'' THEN
STR_TO_DATE(TRIM(LLA.START_DATETIME),'%Y-%m-%d %H:%i:%s.%f') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s.%f') END FC_INICIO_LLAMADA
, CASE WHEN TRIM(LLA.END_DATETIME)<>'' THEN
STR_TO_DATE(TRIM(LLA.END_DATETIME),'%Y-%m-%d %H:%i:%s.%f') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s.%f') END FC_FIN_LLAMADA
, DE.ID_DEPARTAMENTO_CC
, CASE WHEN UPPER(TRIM(LLA.FLG_TRANSFER)) = 'TRUE' THEN 1 ELSE 0 END
FLG_TRANSFERIDO
, AGE.ID_AGENTE_CC
, NOW() FC_INSERT
, STR_TO_DATE('9998-12-31 00:00:00','%Y-%m-%d %H:%i:%s') FC_MODIFICA
FROM STAGE.STG_CONTACTOS_IVR LLA
INNER JOIN ODS.ODS_DM_DEPARTAMENTOS_CC DE ON CASE WHEN
TRIM(LLA.SERVICE)<>'' THEN UPPER(TRIM(LLA.SERVICE)) ELSE 'DESCONOCIDO'
END = DE.DE_DEPARTAMENTO_CC
INNER JOIN ODS.ODS_DM_AGENTES_CC AGE ON CASE WHEN TRIM(LLA.AGENT)<>''
THEN UPPER(TRIM(LLA.AGENT)) ELSE 'DESCONOCIDO' END = AGE.DE_AGENTE_CC;

COMMIT;
```

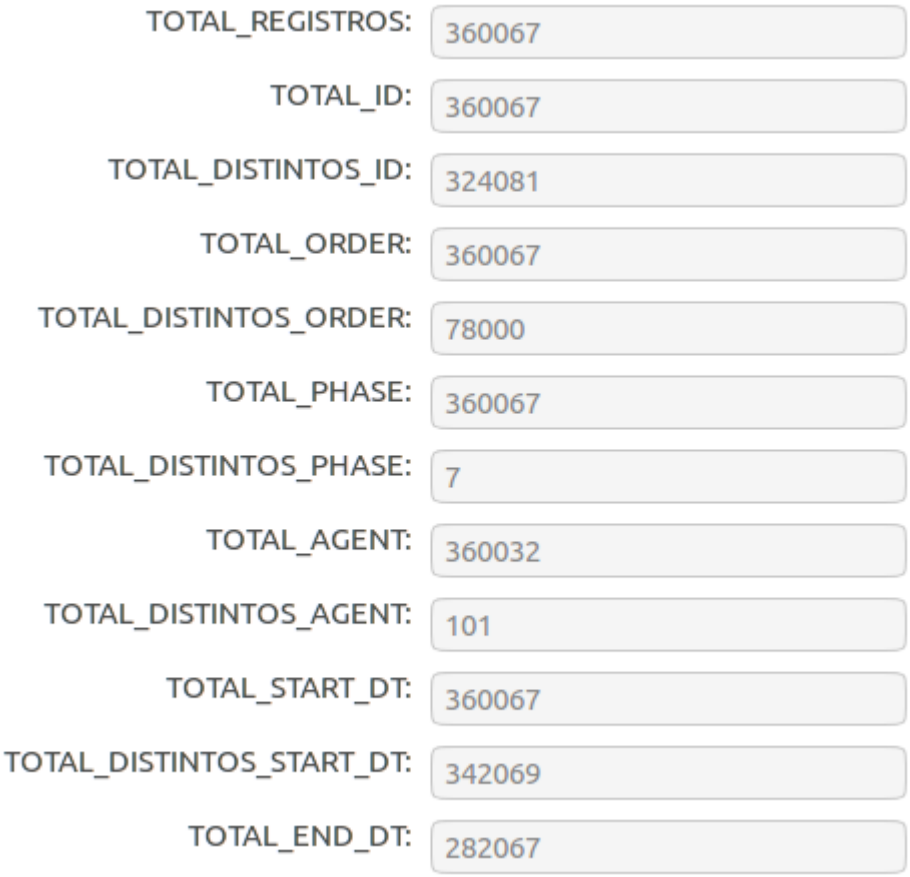
d) Estudio de la tabla STG_ORDERS_CRM:

Analisis STG_ORDERS_CRM.sql

```
USE STAGE;

SELECT COUNT(*) TOTAL_REGISTROS
, SUM(CASE WHEN LENGTH(TRIM(ID)) <> 0 THEN 1 ELSE 0 END) AS TOTAL_ID
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(ID)) <> 0 THEN ID ELSE 0 END) AS
TOTAL_DISTINTOS_ID
, SUM(CASE WHEN LENGTH(TRIM(`ORDER`)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_ORDER
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(`ORDER`)) <> 0 THEN `ORDER` ELSE
0 END) AS TOTAL_DISTINTOS_ORDER
, SUM(CASE WHEN LENGTH(TRIM(PHASE)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_PHASE
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(PHASE)) <> 0 THEN PHASE ELSE 0
END) AS TOTAL_DISTINTOS_PHASE
, SUM(CASE WHEN LENGTH(TRIM(AGENT)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_AGENT
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(AGENT)) <> 0 THEN AGENT ELSE 0
END) AS TOTAL_DISTINTOS_AGENT
, SUM(CASE WHEN LENGTH(TRIM(START_DT)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_START_DT
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(START_DT)) <> 0 THEN START_DT
ELSE 0 END) AS TOTAL_DISTINTOS_START_DT
, SUM(CASE WHEN LENGTH(TRIM(END_DT)) <> 0 THEN 1 ELSE 0 END) AS
TOTAL_END_DT
, COUNT(DISTINCT CASE WHEN LENGTH(TRIM(END_DT)) <> 0 THEN END_DT ELSE 0
END) AS TOTAL_DISTINTOS_END_DT
FROM STAGE.STG_ORDERS_CRM;
```

Esta consulta nos da como resultado:



Conclusiones:

- El ID no es único, aunque no aparecen nulos a veces repite valor, por lo que no podría ser Primary Key de la tabla que creemos y además esta circunstancia es un poco extraña, porque a priori parece el campo que nos identificaría unívocamente cada registro de ORDERS. No será nuestra Primary Key, por lo que crearemos un ID autogenerado para la tabla de PROVISIONES de ODS.
- Observando el campo ORDER, parece que los valores se corresponden con los valores del ID_SERVICIO de nuestra tabla ODS_HC_SERVICIOS. Hago el cruce oportuno y compruebo que efectivamente los campos cruzan, pero hay 78000 valores de ORDER distintos y 78495 ID_SERVICIO. Hago la siguiente select:

  ```
  SELECT SER.ID_SERVICIO
  FROM ODS.ODS_HC_SERVICIOS SER
  LEFT OUTER JOIN STAGE.STG_ORDERS_CRM ORD ON SER.ID_SERVICIO =
  CAST(ORD.ORDER AS SIGNED INTEGER)
  WHERE ORD.ID IS NULL;
  ```

  Y veo que hay 495 servicios que no tienen correspondencia en la tabla ORDER. Nos encontramos una vez más con errores que hemos podido heredar del operacional y que habría que intentar corregir para que los datos fueran coherentes.
- Hay 7 valores distintos para PHASE, por lo que crearemos una dimensión para este campo.

- De nuevo nos encontramos con el campo AGENT, en este caso relleno con una cadena que parece el nombre del usuario del agente, al igual que nos pasaba con los agentes de la tabla LLAMADAS. También encontrábamos este campo en la tabla SERVCIOS, relleno con un código, pero no podemos establecer relación entre ese código y el campo AGENT de ODERS, ya que por un lado en la tabla SERVICIOS aparece un ID_AGENTE y luego cada una de las fases por las que pasa el servicio (desglosada en la tabla ORDERS) tiene un agente diferente. Crearé una dimensión para este campo que guardaré en la tabla ODS_DM_AGENTES_PROV.
- A la cuestión de si realmente nos interesa trasladar esta tabla a nuestro DataWarehouse yo creo que sí es interesante tener esta información, ya que permite bajar a otro nivel de detalle dentro de cada SERVICIO.

1. Creamos las tablas del modelo PROVISIONES:

Creacion modelo provisiones.sql

```
USE ODS;

DROP TABLE IF EXISTS ODS_DM_FASES;

CREATE TABLE ODS_DM_FASES
(ID_FASE INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_FASE VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);

DROP TABLE IF EXISTS ODS_DM_AGENTES_PROV;

CREATE TABLE ODS_DM_AGENTES_PROV
(ID_AGENTE_PROV INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, DE_AGENTE_PROV VARCHAR(512)
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);

DROP TABLE IF EXISTS ODS_HC_PROVISIONES;

CREATE TABLE ODS_HC_PROVISIONES
(
ID_PROVISION INT UNSIGNED AUTO_INCREMENT PRIMARY KEY
, ID INT(11)
, ID_SERVICIO INT(11)
, ID_FASE INT(11)
, ID_AGENTE_PROV INT(10) UNSIGNED
, FC_INICIO DATETIME
, FC_FIN DATETIME
, FC_INSERT DATETIME
, FC_MODIFICACION DATETIME
);
```

2. Creamos las Foreing Keys:

FK modelo provisiones.sql

```
USE ODS;
```

```
ALTER TABLE ODS.ODS_HC_PROVISIONES ADD INDEX FK_PROV_SER_IDX
(ID_SERVICIO ASC);
ALTER TABLE ODS.ODS_HC_PROVISIONES ADD CONSTRAINT FK_PROV_SER FOREIGN
KEY (ID_SERVICIO)
  REFERENCES ODS.ODS_HC_SERVICIOS (ID_SERVICIO);

ALTER TABLE ODS.ODS_HC_PROVISIONES ADD INDEX FK_FASE_IDX (ID_FASE ASC);
ALTER TABLE ODS.ODS_HC_PROVISIONES CHANGE COLUMN ID_FASE ID_FASE INT(10)
UNSIGNED;
ALTER TABLE ODS.ODS_HC_PROVISIONES ADD CONSTRAINT FK_FASE FOREIGN KEY
(ID_FASE)
  REFERENCES ODS.ODS_DM_FASES (ID_FASE);

ALTER TABLE ODS.ODS_HC_PROVISIONES ADD INDEX FK_PROV_AGEN_IDX
(ID_AGENTE_PROV ASC);
ALTER TABLE ODS.ODS_HC_PROVISIONES ADD CONSTRAINT FK_PROV_AGEN FOREIGN
KEY (ID_AGENTE_PROV)
  REFERENCES ODS.ODS_DM_AGENTES_PROV (ID_AGENTE_PROV);
```

3. Poblamos el modelo Provisiones:

POBLAR_ODS_HC_PROVISIONES.sql

```
USE ODS;

INSERT INTO ODS.ODS_DM_FASES (DE_FASE, FC_INSERT, FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(PHASE)), NOW(), NOW()
FROM STAGE.STG_ORDERS_CRM
WHERE TRIM(PHASE)<>'';

COMMIT;

INSERT INTO ODS.ODS_DM_FASES VALUES (98, 'NO APLICA', NOW(), NOW());
INSERT INTO ODS.ODS_DM_FASES VALUES (99, 'DESCONOCIDO', NOW(), NOW());

COMMIT;

INSERT INTO ODS.ODS_DM_AGENTES_PROV (DE_AGENTE_PROV, FC_INSERT,
FC_MODIFICACION)
SELECT DISTINCT UPPER(TRIM(AGENT)),
NOW(), NOW()
FROM STAGE.STG_ORDERS_CRM
WHERE TRIM(AGENT)<>'';

COMMIT;

INSERT INTO ODS.ODS_DM_AGENTES_PROV VALUES (998, 'NO APLICA', NOW(),
NOW());
INSERT INTO ODS.ODS_DM_AGENTES_PROV VALUES (999, 'DESCONOCIDO', NOW(),
NOW());

COMMIT;

INSERT INTO ODS.ODS_HC_PROVISIONES (ID, ID_SERVICIO, ID_FASE,
ID_AGENTE_PROV, FC_INICIO, FC_FIN, FC_INSERT, FC_MODIFICACION)
SELECT CAST(ID AS SIGNED INTEGER) ID
, CAST(`ORDER` AS SIGNED INTEGER) ID_SERVICIO
, ID_FASE
, ID_AGENTE_PROV
, CASE WHEN TRIM(ORD.START_DT)<>'' THEN
STR_TO_DATE(TRIM(ORD.START_DT),'%Y-%m-%d %H:%i:%s.%f') ELSE
STR_TO_DATE('9999-12-31','%Y-%m-%d %H:%i:%s.%f') END FC_INICIO
```

```
, CASE WHEN TRIM(ORD.END_DT)<>'' THEN STR_TO_DATE(TRIM(ORD.END_DT),'%Y-
%m-%d %H:%i:%s.%f') ELSE STR_TO_DATE('9999-12-31','%Y-%m-%d
%H:%i:%s.%f') END FC_FIN
, NOW() FC_INSERT
, STR_TO_DATE('9998-12-31 00:00:00','%Y-%m-%d %H:%i:%s') FC_MODIFICA
FROM STAGE.STG_ORDERS_CRM ORD
INNER JOIN ODS.ODS_DM_FASES FA ON CASE WHEN TRIM(ORD.PHASE)<>'' THEN
UPPER(TRIM(ORD.PHASE)) ELSE 'DESCONOCIDO' END = FA.DE_FASE
INNER JOIN ODS.ODS_DM_AGENTES_PROV AGE ON CASE WHEN TRIM(ORD.AGENT)<>''
THEN UPPER(TRIM(ORD.AGENT)) ELSE 'DESCONOCIDO' END = AGE.DE_AGENTE_PROV;

COMMIT;
```

Segunda parte:

1. Adjuntar el diagrama de ODS completo con el número de registros que contiene cada tabla: Antes de hacer el diagrama voy a crear los registros en la tabla ODS_HC_CLIENTES que aparecían en las tablas STAGE.STG_PRODUCTOS_CRM y STAGE.STG_CONTACTOS_IVR (en los scripts de creación de las Foreign Keys de estas dos tablas comenté las líneas que creaban la relación con la tabla de clientes para que no me diera error al insertar los datos del operacional). Después de insertar estos datos ejecuto las líneas comentadas de ambos scripts para tener todas las relaciones y por último crearé el diagrama de ODS.

   Contenido del script Clientes desconocidos.sql:

```
INSERT INTO ODS_HC_CLIENTES
SELECT DISTINCT(FAC.ID_CLIENTE), 'DESCONOCIDO', 'DESCONOCIDO','99-999-
9999', 99, 999999,9999999999,
'DESCONOCIDO@DESCONOCIDO.ES',STR_TO_DATE('31/12/9999','%d/%m/%Y'),999,99
9,NOW(),STR_TO_DATE('31/12/9999','%d/%m/%Y')
FROM ODS_HC_FACTURAS FAC
LEFT OUTER JOIN ODS_HC_CLIENTES CLI ON FAC.ID_CLIENTE = CLI.ID_CLIENTE
WHERE CLI.ID_CLIENTE IS NULL;

COMMIT;

INSERT INTO ODS_HC_CLIENTES VALUES (999910000,'DESCONOCIDO',
'DESCONOCIDO','99-999-9999', 99, 999999,9999999999,
'DESCONOCIDO@DESCONOCIDO.ES',STR_TO_DATE('31/12/9999','%d/%m/%Y'),999,99
9,NOW(),STR_TO_DATE('31/12/9999','%d/%m/%Y'));
COMMIT;
```

   El diagrama del operacional está en el archivo: **Modelo ODS.mwb**

   Número de registros de cada tabla:
   - ODS_DM_AGENTES_CC: 595, tiene sentido porque al hacer el estudio de la tabla STG_CONTACTOS_IVR veíamos 594 valores distintos para el campo AGENT, uno de ellos se corresponde a la cadena vacía, que yo traduje en 'DESCONOCIDO' y otro más que añadí para el valor 'NO APLICA'.

   - ODS_DM_AGENTES_PROV: 102, la explicación es la misma, 101 valores distintos contando con la cadena vacía, que sería el 'DESCONOCIDO' y un valor más para el 'NO APLICA'.

- ODS_DM_CANALES: 6, como en los dos casos anteriores, había 5 valores distintos incluyendo la cadena vacía, y añadimos uno más que es el 'NO APLICA'.

- ODS_DM_CICLOS_FACTURACION: 4, en este caso había 2 ciclos de facturación distintos, pero estaban todos los registros rellenos, por lo que añado dos valores más uno para 'DESCONOCIDO' y otro para 'NO APLICA'.

- ODS_DM_CIUDADES_ESTADO: 158. Este resultado tiene sentido, porque en la tabla de STG_CLIENTES_CRM hay 83 valores distintos para las tuplas CITY y STATE y en la tabla STG_PRODUCTOS_CRM hay 157 valores distintos. Para ver si todos los valores que están en STG_CLIENTES_CRM están también en la tabla STG_PRODUCTOS_CRM hago la siguiente consulta, que no me devuelve ningún resultado:

```
SELECT DISTINCT CITY, STATE
FROM STAGE.STG_CLIENTES_CRM
WHERE (CITY, STATE) NOT IN
                  (SELECT DISTINCT PRODUCT_CITY, PRODUCT_STATE
                     FROM STAGE.STG_PRODUCTOS_CRM);
```

  Por lo que los 83 valores de la tabla STG_CLIENTES_CRM están incluidos en los 157 valores para ciudad – estado de la tabla STG_PRODUCTOS_CRM, entre los que se encuentra el valor para la cadena vacía ('DESCONOCIDO') y yo añado uno más para el 'NO APLICA', en total 158 registros.

- ODS_DM_COMPANYAS: 385. Del estudio que se realizó en primer lugar de la tabla STG_CLIENTES_CRM se concluía que había 384 valores distintos para el campo COMPANY, más el 'NO APLICA', en total 385.

- ODS_DM_DEPARTAMENTOS_CC: 8. En el estudio de la tabla STG_CONTACTOS_IVR vimos que había 7 valores distintos para el campo SERVICE del que sacamos la dimensión de DEPARTAMENTOS, más el 'NO APLICA', en total 8 valores.

- ODS_DM_FASES: 9. Parece correcto, ya que veíamos 7 valores distintos del campo PHASE en la tabla STG_ORDERS_CRM entre los que no se encontraba la cadena vacía, por lo que se añaden dos registros más, correspondientes a los valores 'DESCONOCIDO' y 'NO APLICA'.

- ODS_DM_METODOS_PAGO: 5. Es correcto ya que había 3 valores distintos más el 'DESCONOCIDO' y 'NO APLICA', total 5 registros.

- ODS_DM_PAISES: 3 registros. Si recordamos había dos valores diferentes para el mismo país, 'US' y 'United States', que decidimos transformarlo en el único valor 'US' ya que ambos se refieren al mismo país. A este valor le añadimos los dos valores de 'DESCONOCIDO' y 'NO APLICA' y nos salen los tres registros.

- ODS_DM_PRODUCTOS: 8 registros. Corresponden a los 6 valores diferentes del campo PRODUCT_NAME de la tabla STG_PRODUCTOS_CRM, entre los que no se encontraba la cadena vacía. Añadimos como siempre los dos registros para los valores 'DESCONOCIDO' y 'NO APLICA'.

- ODS_DM_PROFESIONES: 197 registros. En el estudio que se hizo de la tabla STG_CLIENTES_CRM veíamos 196 valores para el campo PROFESION, incluyendo la cadena vacía, por lo que añadimos un registro más para el 'NO APLICA' y nos salen los 197 registros.

- ODS_DM_SEXOS: 4 registros, 3 valores que aparecían en el estudio de STG_CLIENTES_CRM, que eran MALE, FEMALE y la cadena vacía que se tradujo en 'DESCONOCIDO'. Añadimos uno más para 'NO APLICA'.

- ODS_HC_CLIENTES: 20002. Veíamos que en la tabla STG_CLIENTES_CRM había un total de 17558 registros. A estos registros le añadimos posteriormente aquellos clientes que aparecían en la tabla STG_FACTURAS_CRM, que eran un total de 2442 y un cliente más que aparecía en la tabla STG_PRODUCTOS_CRM (si recordamos en la tabla de productos aparecían 3 clientes que no estaban en la tabla clientes, pero dos de ellos eran comunes a la tabla de facturas). En total entonces tenemos 17558 + 2442 + 1 = 20001. Al hacer el estudio de la tabla STG_CONTACTOS_IVR no pudimos establecer una relación entre el PHONE_NUMBER y el ID_CLIENTE, por lo que decidí rellenar este campo con el valor 999999999, para indicar 'DESCONOCIDO', por eso introduje un campo más en la tabla de clientes y de ahí que haya 20002. Podría haber modificado el modelo que nos mandaste por pdf y haber eliminado la relación de la tabla LLAMADAS con la tabla de CLIENTES, quitando el campo ID_CLIENTE, pero me pareció más correcto dejar reflejada la relación, aunque sea de forma 'ficticia', para luego más adelante en la parte de Data Management proponer como mejora un campo ID_CLIENTE en la tabla CONTACTOS que se rellene de forma adecuada y que nos dé información del cliente que contacta con nosotros.

- ODS_HC_DIRECCIONES: 95769 registros. Tiene sentido porque había 17498 direcciones distintas en la tabla CLIENTES y 78271 direcciones distintas en la tabla PRODUCTOS que no estaban en CLIENTES, 17498 + 78271 = 95769.

- ODS_HC_FACTURAS: 420000 registros. Son los registros que aparecían en la tabla STG_FACTURAS_FCT.

- ODS_HC_LLAMADAS: 202717 registros. Corresponden a los registros de la tabla STG_CONTACTOS_IVR.

- ODS_HC_PROVISIONES: 360067 registros, que son los que nos encontramos en la tabla STG_ORDERS_CRM.

- ODS_HC_SERVICIOS: 78495 registros, que son los que tiene la tabla STG_PRODUCTOS_CRM.

2. ¿Por qué en el modelo de DIRECCIONES dejo en la misma tabla las CIUDADES y los ESTADOS y no los separo en dos tablas distintas? Porque al hacer esto nos damos cuenta que al comprobar el número de registros totales nos da 84, mientras que el número de registros distintos es 83. Mirando más a fondo vemos que hay una ciudad, Glendale, con el mismo nombre en dos estados distintos, Arizona y California:

```
SELECT DISTINCT A.CITY, A.STATE, COUNT(*)
FROM (
SELECT *
FROM STAGE.STG_CLIENTES_CRM CLI) A
GROUP BY A.CITY, A.STATE
```

3. OPCIONAL: Separar el campo DE_DIRECCION en dos campos, NOMBRE_VIA y NUM_VIA. **Segunda parte opcional.sql**:

```
SELECT CASE WHEN DE_DIRECCION = 'DESCONOCIDO' THEN 9999 ELSE CASE WHEN
DE_DIRECCION = 'NO APLICA' THEN 9998 ELSE SUBSTRING_INDEX(DE_DIRECCION,'
',1) END END NUM_VIA
, CASE WHEN DE_DIRECCION = 'DESCONOCIDO' THEN 'DESCONOCIDO' ELSE CASE
WHEN DE_DIRECCION = 'NO APLICA' THEN 'NO APLICA' ELSE
SUBSTR(DE_DIRECCION,LOCATE(' ',DE_DIRECCION,1)) END END NOMBRE_VIA
FROM ODS_HC_DIRECCIONES;
```

Tercera parte:

1. Qué habrías echo diferente centrándote en las “patas”:

   - Data Quality: Definir, mejorar y controlar la calidad de los datos. En este ejemplo se podrían hacer varias cosas para conseguir esto:
     - Tabla de Clientes:
       i. Deben estar registrados todos los clientes de la compañía con nombre, apellidos, número de documento, dirección y teléfono como datos que tienen que ser válidos y deben estar actualizados.
     - Tabla Productos:
       i. Definir un periodo máximo en el que pueden aparecer valores nulos en INSTALL_DATE y END_DATE en un mismo registro, pasado ese periodo al menos una de estas dos fechas debe estar rellena y si no es así habría que añadir un campo que nos informe que ese servicio no ha sido llevado a cabo finalmente.
       ii. No pueden venir vacíos los campos que recogen la información de la dirección del servicio en aquellos servicios que se refieran a productos que necesitan una dirección física en la que instalarlos.

     - Tabla de Facturas:
       i. No puede haber CUSTOMER_ID que no aparezcan en la tabla de CLIENTES.

- Tabla de Orders:
  i. No puede haber valores en la tabla de PRODUCTOS que no estén reflejados en la tabla de ORDERS. Veíamos que había 495 registros en la tabla PRODUCTOS que no tenían correspondencia con registros de la tabla ORDERS.

- Master Data:
  Podríamos establecer como tablas maestras la tabla de direcciones, ya que tanto CLIENTES como PRODUCTOS tiran de esta tabla. Al estar en una tabla maestra evitamos que se produzcan cosas como lo que nos ocurría en el caso de país, que en un sitio se representaba como US y en otro como United States.
  También podríamos crear una tabla maestra de agentes, ya que tanto PRODUCTOS como CONTACTOS y ORDERS tienen un campo que informa del agente. Estaría bien tener una tabla maestra con los agentes de la compañía de la que tiraran las tres tablas y usar ese ID en los tres casos (ahora se utiliza un código en un sitio y un login en los otros).

- Data Modeling & Design:
  - Tablas del CRM:
    Clientes:
    i. Customer_id que sea un entero y Primary Key.
    ii. First Name, Last Name, Identified Doc, City, Address, State, Country campos no nulos.
    iii. Postal Code y Phone también campos no nulos y además numéricos en lugar de cadenas.

    (*) Si usamos tablas maestras para la dirección en realidad todos los campos relacionados con ésta deberían desaparecer y tener un Id de dirección que sea Foreing Key a la tabla maestra.

    Productos:

    i. Product_id que sea un entero y Primary Key.
    ii. Customer_Id debe ser una Foreing Key del Customer_Id de la tabla clientes.
    iii. Start_Date, Install_Date, End_Date deben ser campos de tipo fecha.
    iv. Agent_Code y Postal_Code enteros.

    (*) Si usamos tablas maestras para la dirección en realidad todos los campos relacionados con ésta deberían desaparecer y tener un Id de dirección que sea Foreing Key a la tabla maestra.

    Orders:
    i. El ID debería ser la clave primaria, no permitiéndose valores nulos en ese campo.
    ii. El campo ORDER tiene que ser una Foreing Key del campo PRODUCT_ID de la tabla PRODUCTOS, además de ser un campo numérico.

iii. START_DT y END_DT deberían ser campos fecha.

- Tablas de FACTURADOR:
  Facturas:
  i. Bill_Ref_No debería ser numérico y Primary Key.
  ii. Customer_Id debe ser una Foreing Key del Customer_Id de la tabla clientes. Como normalmente las tablas están en BBDD separadas esto no va a ser posible, más adelante en el siguiente punto de la práctica explico cómo sería posible aplicando las disciplinas del Data Governance.
  iii. Start_Date, End_Date, Statement_Date y Payment_Date deben ser campos fecha.
  iv. Amount debería ser un decimal y no nulo.

- Tablas de IVR:
  Contactos:
  i. Un nuevo campo con el ID_CLIENTE, que debe estar en concordancia con el Customer_Id de la tabla CLIENTES del CRM.
  ii. El ID debería ser un campo numérico, al igual que el Phone_number.
  iii. Start_Datetime y End_DateTime deben ser campos fecha.
  iv. Los campos Agent y Services deberían ser no nulos.

2. ¿Aconsejarías algún cambio en los sistemas de origen extra teniendo en cuenta el resto de disciplinas del Data Governance?
   - Necesitamos un acceso global a los datos del cliente por todas las partes de nuestra organización: CRM, FACTURADOR e IVR. No sólo por la relación de Foreing Keys de las que hablaba antes, además podría ser interesante para tener los datos del cliente actualizados, que cuando un cliente se pone en contacto con el call center verificar que los datos de contacto del cliente son los que tenemos guardados en nuestro sistema, de no ser así, deberá actualizarse la tabla de CLIENTES con los nuevos datos.

Cuarta Parte:

1. ¿Plantearías otro diseño mejorado?

   - Además de STAGE, ODS y DDS, podríamos tener otra BBDD más, ODS-DET, donde guardaríamos las tablas temporales en las que nos tenemos que apoyar para rellenar las tablas de ODS. En este caso sería las tablas TMP_DIRECCIONES.
   - También puede ser interesante tener otra BBDD para generar informes de errores (tablas que se rellenen con los errores que hemos visto al analizar las tablas anteriores), tablas que contengan información de las instalaciones pendientes para sacar un listado diario que se les pasaría a los técnicos de las cosas que tienen que hacer ese día… etc…
   - Y una BBDD para realizar analíticas: medias de instalaciones, bajas, nuevas altas… etc…

Quinta Parte:

1. Reglas o mandamientos de un DataWarehouse:
    - Cargar los datos del operacional tal cual nos vienen en STAGE, para tener una foto exacta de los datos que nos llegan y a partir de ahí realizar las modificaciones que necesitemos.
    - El DataWarehouse reúne todos los datos de la organización en un mismo entorno en el que se guarda la información de una forma relacionada.
    - Se guardan también históricos de la información que nos permiten realizar un análisis de los datos para tomar decisiones.
    - Necesitamos sistemas potentes que puedan mover mucha cantidad de datos.
    - Eliminar la aparición de valores nulos para los campos, sustituyéndolos por los correspondientes 'NO APLICA' y 'DESCONOCIDO'.
    - Pasar a tablas de dimensiones todas aquellas cadenas que se repiten continuamente en un campo.

Sexta Parte:

- ¿Nivel de SQL antes y después? Pues partí pensando que me encontraba en un nivel 'Todo lo que quiso saber sobre SQL y no se atrevió a preguntar', y en el desarrollo del Track me he dado cuenta de que más bien me encontraba en un punto intermedio entre 'Estoy más perdido que un muelle en las escaleras de Howarts' y 'Dejad que las querys se acerquen a mi'. El nivel ha subido después de la práctica, yo diría que ya puedo afirmar que estoy en el nivel 'Dejad que las querys se acerquen a mi' avanzado, pero soy consciente de que me queda mucho que practicar para estar suelta en todo el ecosistema de DataWarehouse, aunque estoy muy contenta con el nivel que he alcanzado. Creo que con este módulo he asentado una base sólida de conocimiento.
- ¿Algún comentario extra que quieras hacer? Agradecerte la dedicación y las horas que has estado preparando todo, el habernos dado todos los pasos explicados lo mejor posible y también la decisión de dejar la entrega de la práctica para más adelante. Esto ha sido fundamental para poder digerir todo el conocimiento. Si hubiese sido de otra forma habría entregado cualquier cosa hecha rápido y pronto, y, aunque seguro que la práctica es muy mejorable, por lo menos he tenido tiempo para desarrollar los puntos lo mejor que he sabido.