# Option #1: Data Loading - Northwind Data

Scott Miner

Colorado State University – Global Campus

## Option #1: Data Loading - Northwind Data

In his final Portfolio Project for *MIS541-Data Warehousing in Enterprise Environments*, the student will extract data from the Northwind OLTP system, transform the data to conform to the star schema model developed in *Portfolio Milestones 1 and 2*, and finally load the modified data into the Northwind Data Warehouse. After populating the destination tables with the transformed data, the student will provide SQL code that returns the row counts of each populated table, along with a listing of the first ten rows of each table. Lastly, the student will provide a brief synopsis of the lessons learned from his Portfolio Project, along with any recommendations he might give to an organization that plans to develop an enterprise data warehouse.

In this paper, the student models the *order fulfillment* business process. The *grain* of the fact table is one row per order transaction. What are some of the business questions such a dimensional model can answer? These questions include, “What are the total product quantities sold in a given year by quarter and product?” “Which employees, products, and customers generate the most revenue?” and “For a given year, what are our most expensive products to ship and which carriers ship them most often?” The business process, the grain of the fact table, and the business questions, therefore, remain the same between Modules 3, 5, and 8.

The student did, however, incorporate changes into the star schema design. Upon the initial load of the **EMPLOYEE** dimension, the student realized the dimension contained a self-referential constraint, where the *reports_to* field of the table referenced the *employee_id* field of each employee’s supervisor for every employee that had one. Kimball and Ross (2013) inform us that a dimensional model that represents a ragged variable depth hierarchy via recursive pointers, as witnessed here, potentially creates a problem because the tree structure representation is entangled with the dimension (Kimball & Ross, 2013).

The solution, suggested by Kimball and Ross (2013), is to build a bridge table that contains all the information about the hierarchical relationship. The grain of the bridge table is one row for "each possible parent to each possible child, including a row that connects the parent to itself" (Kimball & Ross, 2013, p. 217). The student constructed this bridge table manually (without SQL code), using the samples provided by Kimball and Ross and McHugh (2017a) for reference. Figure 22, presented at the end of this paper, provides a listing of the first ten rows of this bridge table. What additional business questions does the bridge table allow us to answer? These questions include the following: "How many people work for a given manager?" "What are the total sales revenues for the employees who report to a given manager, broken down by product?" and "What is the hierarchy of management to whom a given employee reports?" (McHugh, 2017b). As we can see, the **MANAGEMENT HIERARCHY** bridge table is beneficial for reporting rollup facts related to the hierarchy of management. The use of the bridge table, however, is not mandatory when analyzing facts solely related to employees. Analysts may choose to by-pass the bridge table altogether, joining the **EMPLOYEE** dimension directly to the **ORDER DETAIL** fact table when reports do not call for metrics regarding hierarchical management relationships. The advantages of bridge tables for ragged hierarchies include limited impacts when nodes and tree structures change (Kimball & Ross, 2013).

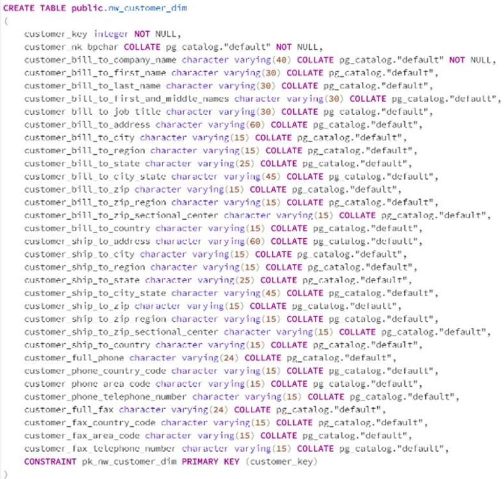
Another change to the star schema design involves the *territory* and *region* attributes of the data warehouse. Initially, the student planned to incorporate these attributes into the **EMPLOYEE** dimension. Upon further analysis, however, it turns out these attributes are not tied directly to the **ORDERS** table in the transactional system, as the **EMPLOYEES**,

**CUSTOMERS**, **SHIPPERS**, and **PRODUCTS** tables are, but to the **EMPLOYEES** table. Each employee serves multiple territories. Each employee is linked to a single transaction. There is, therefore, no way to determine to which territory a particular transaction is linked. We can, however, learn more about territorial and regional sales metrics if we migrate these attributes to the data warehouse. Because each employee is linked to a single transaction, and each employee serves multiple territories, this scenario presents us with a perfect circumstance to model *multivalued dimensions,* where a dimension takes on multiple values at the fact table's grain (Kimball & Ross, 2013).

To model these multivalued dimensions, the student followed the advice of Kimball and Ross (2013) utilizing a group bridge table, which provides two columns, one for the group dimension key and another for each employee's territories grouped into a concatenated list. Figure 26 displays a listing of this table's contents. The group table links both to the **EMPLOYEE** dimension and the **TERRITORY BRIDGE** table. The **TERRITORY BRIDGE** table, in turn, links to the **TERRITORY** dimension, which contains columns for each of the 53 territories and their associated regions. Forty-nine of these territories are associated with employees. The **TERRITORY BRIDGE** table reflects this count. As employees are assigned additional territories, we can add the corresponding rows and values to the bridge tables. Modeling these multivalued dimensions allows us to answer additional business questions such as, "In which territories and regions are revenues highest?"

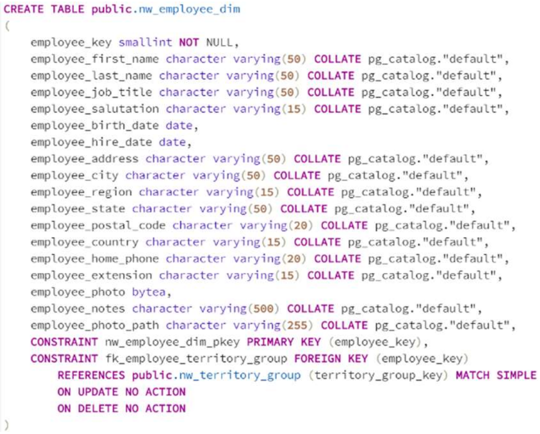
What are the lessons the student learned from this portfolio project? For one, the student learned to create a dimensional model based on a transactional system. The goals of a dimensional model are to deliver data that are understandable to business users and deliver fast query performance. Star schemas present us with a perfect design architecture to obtain such goals. Dimensions in star schemas are often denormalized, storing hierarchical descriptive information redundantly. Depending on the goals of the business and underlying data structure, we may incorporate bridge tables into the design to allow analysts to report on hierarchical relationships and multivalued dimensions. Therefore, the student recommends that organizations gain a clear understanding of the business needs and underlying source data early in the design process. Additionally, the student recommends that businesses follow the four-step dimensional design process as outlined by Kimball and Ross (2013): (a) Select the business process, (b) Declare the grain, (c) Identify the dimensions, and (d) Identify the facts. Incorporating conformed dimensions into the data warehouse allows organizations to decrease the time-to-market for future deliverables and leads to agile decision making. By implementing these lessons, being open to feedback, and understanding the potential value a data warehouse offers, the student was able to implement a successful ETL solution for Northwind Traders.

In conclusion, this paper presented the student's process for designing the Northwind Data Warehouse. The business process, the grain of the fact table, and the business questions all remained the same between Modules 3, 5, and 8. The student did incorporate changes to the star schema design, adding bridge tables to model the hierarchy of management relationships and multivalued dimensions. Lastly, this paper presents a brief synopsis of the lessons the student learned and recommendations to future organizations. On the following pages, Figures 1 – 10 display the SQL code to create each warehouse table; Figures 11 – 17 display the SQL code to extract the needed variables from the transactional system and perform any necessary transformations; Figures 18 – 28 display the first ten rows of each table. Lastly, Figures 29 – 32 display the SQL code to load each table, the row counts of each table, the updated star schema, and the SQL code to generate the table listings.



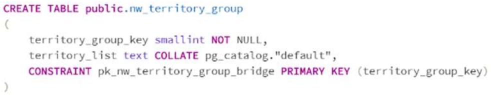
```
CREATE TABLE public.nw_customer_dim
(
    customer_key integer NOT NULL,
    customer_nk bpchar COLLATE pg_catalog."default" NOT NULL,
    customer_bill_to_company_name character varying(40) COLLATE pg_catalog."default" NOT NULL,
    customer_bill_to_first_name character varying(30) COLLATE pg_catalog."default",
    customer_bill_to_last_name character varying(30) COLLATE pg_catalog."default",
    customer_bill_to_first_and_middle_names character varying(30) COLLATE pg_catalog."default",
    customer_bill_to_job_title character varying(30) COLLATE pg_catalog."default",
    customer_bill_to_address character varying(60) COLLATE pg_catalog."default",
    customer_bill_to_city character varying(15) COLLATE pg_catalog."default",
    customer_bill_to_region character varying(15) COLLATE pg_catalog."default",
    customer_bill_to_state character varying(25) COLLATE pg_catalog."default",
    customer_bill_to_city_state character varying(45) COLLATE pg_catalog."default",
    customer_bill_to_zip character varying(15) COLLATE pg_catalog."default",
    customer_bill_to_zip_region character varying(15) COLLATE pg_catalog."default",
    customer_bill_to_zip_sectional_center character varying(15) COLLATE pg_catalog."default",
    customer_bill_to_country character varying(15) COLLATE pg_catalog."default",
    customer_ship_to_address character varying(60) COLLATE pg_catalog."default",
    customer_ship_to_city character varying(15) COLLATE pg_catalog."default",
    customer_ship_to_region character varying(15) COLLATE pg_catalog."default",
    customer_ship_to_state character varying(25) COLLATE pg_catalog."default",
    customer_ship_to_city_state character varying(45) COLLATE pg_catalog."default",
    customer_ship_to_zip character varying(15) COLLATE pg_catalog."default",
    customer_ship_to_zip_region character varying(15) COLLATE pg_catalog."default",
    customer_ship_to_zip_sectional_center character varying(15) COLLATE pg_catalog."default",
    customer_ship_to_country character varying(15) COLLATE pg_catalog."default",
    customer_full_phone character varying(24) COLLATE pg_catalog."default",
    customer_phone_country_code character varying(15) COLLATE pg_catalog."default",
    customer_phone_area_code character varying(15) COLLATE pg_catalog."default",
    customer_phone_telephone_number character varying(15) COLLATE pg_catalog."default",
    customer_full_fax character varying(24) COLLATE pg_catalog."default",
    customer_fax_country_code character varying(15) COLLATE pg_catalog."default",
    customer_fax_area_code character varying(15) COLLATE pg_catalog."default",
    customer_fax_telephone_number character varying(15) COLLATE pg_catalog."default",
    CONSTRAINT pk_nw_customer_dim PRIMARY KEY (customer_key)
)
```

*Figure 1*. Customer dimension create statement



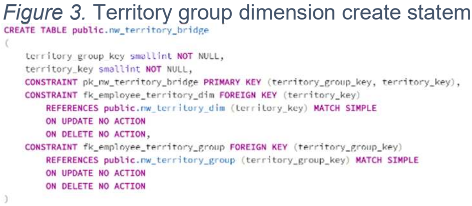
```
CREATE TABLE public.nw_employee_dim
(
    employee_key smallint NOT NULL,
    employee_first_name character varying(50) COLLATE pg_catalog."default",
    employee_last_name character varying(50) COLLATE pg_catalog."default",
    employee_job_title character varying(50) COLLATE pg_catalog."default",
    employee_salutation character varying(15) COLLATE pg_catalog."default",
    employee_birth_date date,
    employee_hire_date date,
    employee_address character varying(50) COLLATE pg_catalog."default",
    employee_city character varying(50) COLLATE pg_catalog."default",
    employee_region character varying(15) COLLATE pg_catalog."default",
    employee_state character varying(50) COLLATE pg_catalog."default",
    employee_postal_code character varying(20) COLLATE pg_catalog."default",
    employee_country character varying(15) COLLATE pg_catalog."default",
    employee_home_phone character varying(20) COLLATE pg_catalog."default",
    employee_extension character varying(15) COLLATE pg_catalog."default",
    employee_photo bytea,
    employee_notes character varying(500) COLLATE pg_catalog."default",
    employee_photo_path character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT nw_employee_dim_pkey PRIMARY KEY (employee_key),
    CONSTRAINT fk_employee_territory_group FOREIGN KEY (employee_key)
        REFERENCES public.nw_territory_group (territory_group_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

*Figure 2*. Employee dimension create statement



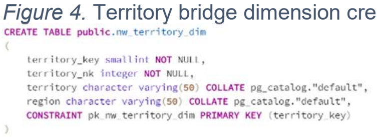
```
CREATE TABLE public.nw_territory_group
(
    territory_group_key smallint NOT NULL,
    territory_list text COLLATE pg_catalog."default",
    CONSTRAINT pk_nw_territory_group_bridge PRIMARY KEY (territory_group_key)
)
```

*Figure 3.* Territory group dimension create statement



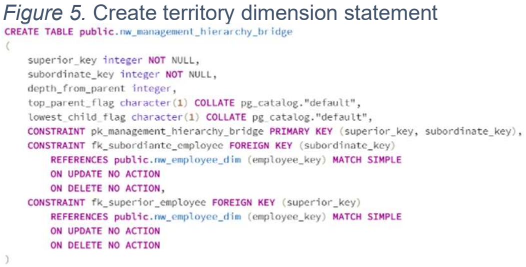
```
CREATE TABLE public.nw_territory_bridge
(
    territory_group_key smallint NOT NULL,
    territory_key smallint NOT NULL,
    CONSTRAINT pk_nw_territory_bridge PRIMARY KEY (territory_group_key, territory_key),
    CONSTRAINT fk_employee_territory_dim FOREIGN KEY (territory_key)
        REFERENCES public.nw_territory_dim (territory_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_employee_territory_group FOREIGN KEY (territory_group_key)
        REFERENCES public.nw_territory_group (territory_group_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

*Figure 4.* Territory bridge dimension create statement



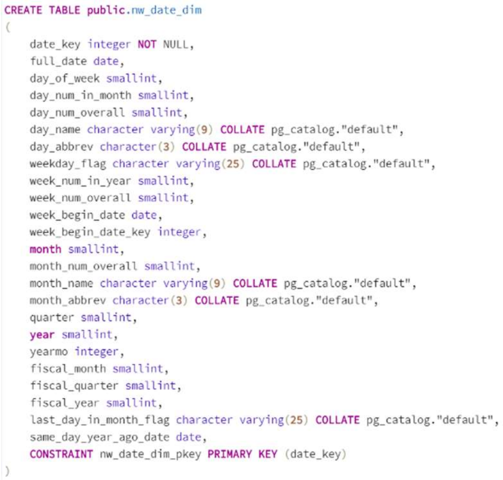
```
CREATE TABLE public.nw_territory_dim
(
    territory_key smallint NOT NULL,
    territory_nk integer NOT NULL,
    territory character varying(50) COLLATE pg_catalog."default",
    region character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT pk_nw_territory_dim PRIMARY KEY (territory_key)
)
```

*Figure 5.* Create territory dimension statement



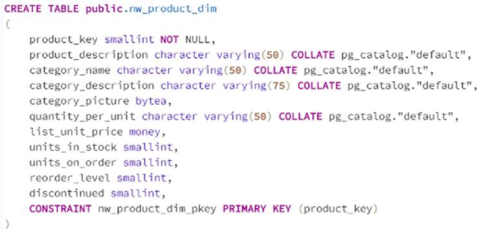
```
CREATE TABLE public.nw_management_hierarchy_bridge
(
    superior_key integer NOT NULL,
    subordinate_key integer NOT NULL,
    depth_from_parent integer,
    top_parent_flag character(1) COLLATE pg_catalog."default",
    lowest_child_flag character(1) COLLATE pg_catalog."default",
    CONSTRAINT pk_management_hierarchy_bridge PRIMARY KEY (superior_key, subordinate_key),
    CONSTRAINT fk_subordiante_employee FOREIGN KEY (subordinate_key)
        REFERENCES public.nw_employee_dim (employee_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_superior_employee FOREIGN KEY (superior_key)
        REFERENCES public.nw_employee_dim (employee_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

*Figure 6.* Management bridge create statement



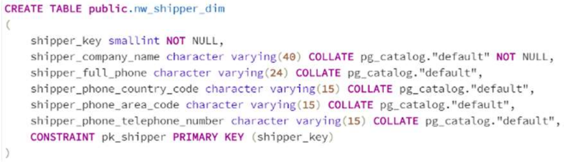
```
CREATE TABLE public.nw_date_dim
(
    date_key integer NOT NULL,
    full_date date,
    day_of_week smallint,
    day_num_in_month smallint,
    day_num_overall smallint,
    day_name character varying(9) COLLATE pg_catalog."default",
    day_abbrev character(3) COLLATE pg_catalog."default",
    weekday_flag character varying(25) COLLATE pg_catalog."default",
    week_num_in_year smallint,
    week_num_overall smallint,
    week_begin_date date,
    week_begin_date_key integer,
    month smallint,
    month_num_overall smallint,
    month_name character varying(9) COLLATE pg_catalog."default",
    month_abbrev character(3) COLLATE pg_catalog."default",
    quarter smallint,
    year smallint,
    yearmo integer,
    fiscal_month smallint,
    fiscal_quarter smallint,
    fiscal_year smallint,
    last_day_in_month_flag character varying(25) COLLATE pg_catalog."default",
    same_day_year_ago_date date,
    CONSTRAINT nw_date_dim_pkey PRIMARY KEY (date_key)
)
```

*Figure 7*. Date dimension create statement



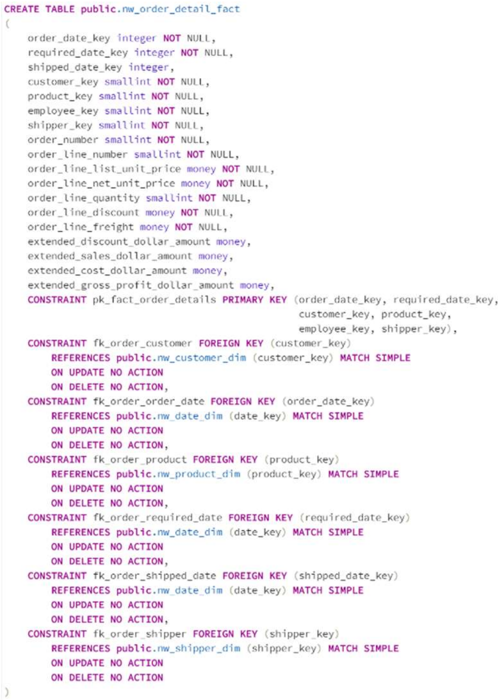
```
CREATE TABLE public.nw_product_dim
(
    product_key smallint NOT NULL,
    product_description character varying(50) COLLATE pg_catalog."default",
    category_name character varying(50) COLLATE pg_catalog."default",
    category_description character varying(75) COLLATE pg_catalog."default",
    category_picture bytea,
    quantity_per_unit character varying(50) COLLATE pg_catalog."default",
    list_unit_price money,
    units_in_stock smallint,
    units_on_order smallint,
    reorder_level smallint,
    discontinued smallint,
    CONSTRAINT nw_product_dim_pkey PRIMARY KEY (product_key)
)
```

*Figure 8*. Product dimension create statement



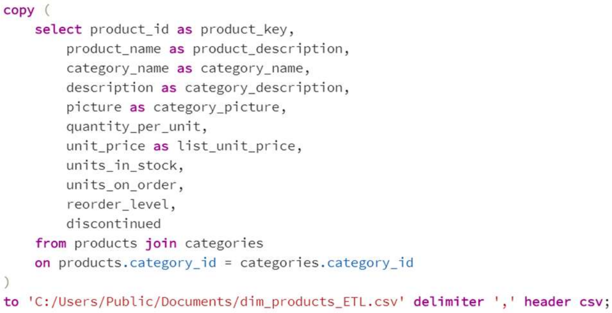
```
CREATE TABLE public.nw_shipper_dim
(
    shipper_key smallint NOT NULL,
    shipper_company_name character varying(40) COLLATE pg_catalog."default" NOT NULL,
    shipper_full_phone character varying(24) COLLATE pg_catalog."default",
    shipper_phone_country_code character varying(15) COLLATE pg_catalog."default",
    shipper_phone_area_code character varying(15) COLLATE pg_catalog."default",
    shipper_phone_telephone_number character varying(15) COLLATE pg_catalog."default",
    CONSTRAINT pk_shipper PRIMARY KEY (shipper_key)
)
```

*Figure 9*. Shipper dimension create statement



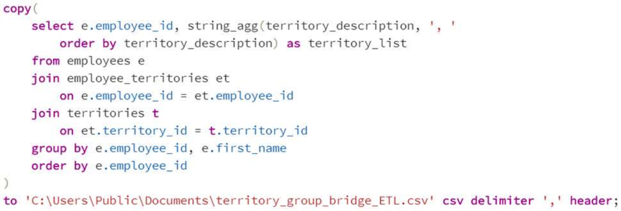
```
CREATE TABLE public.nw_order_detail_fact
(
    order_date_key integer NOT NULL,
    required_date_key integer NOT NULL,
    shipped_date_key integer,
    customer_key smallint NOT NULL,
    product_key smallint NOT NULL,
    employee_key smallint NOT NULL,
    shipper_key smallint NOT NULL,
    order_number smallint NOT NULL,
    order_line_number smallint NOT NULL,
    order_line_list_unit_price money NOT NULL,
    order_line_net_unit_price money NOT NULL,
    order_line_quantity smallint NOT NULL,
    order_line_discount money NOT NULL,
    order_line_freight money NOT NULL,
    extended_discount_dollar_amount money,
    extended_sales_dollar_amount money,
    extended_cost_dollar_amount money,
    extended_gross_profit_dollar_amount money,
    CONSTRAINT pk_fact_order_details PRIMARY KEY (order_date_key, required_date_key,
                                                  customer_key, product_key,
                                                  employee_key, shipper_key),
    CONSTRAINT fk_order_customer FOREIGN KEY (customer_key)
        REFERENCES public.nw_customer_dim (customer_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_order_order_date FOREIGN KEY (order_date_key)
        REFERENCES public.nw_date_dim (date_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_order_product FOREIGN KEY (product_key)
        REFERENCES public.nw_product_dim (product_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_order_required_date FOREIGN KEY (required_date_key)
        REFERENCES public.nw_date_dim (date_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_order_shipped_date FOREIGN KEY (shipped_date_key)
        REFERENCES public.nw_date_dim (date_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_order_shipper FOREIGN KEY (shipper_key)
        REFERENCES public.nw_shipper_dim (shipper_key) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

*Figure 10.* Order detail fact table create statement



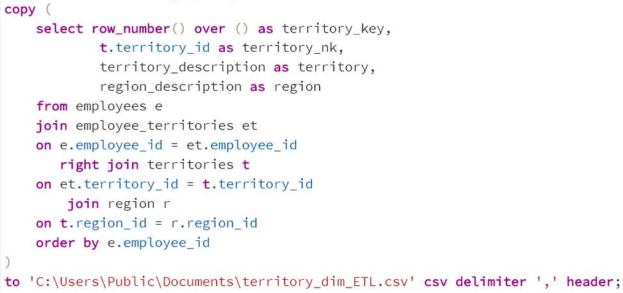
```
copy (
    select product_id as product_key,
        product_name as product_description,
        category_name as category_name,
        description as category_description,
        picture as category_picture,
        quantity_per_unit,
        unit_price as list_unit_price,
        units_in_stock,
        units_on_order,
        reorder_level,
        discontinued
    from products join categories
    on products.category_id = categories.category_id
)
to 'C:/Users/Public/Documents/dim_products_ETL.csv' delimiter ',' header csv;
```

*Figure 11.* Product dimension extract



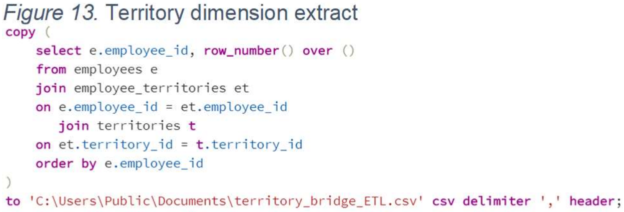
```
copy(
    select e.employee_id, string_agg(territory_description, ', '
        order by territory_description) as territory_list
    from employees e
    join employee_territories et
        on e.employee_id = et.employee_id
    join territories t
        on et.territory_id = t.territory_id
    group by e.employee_id, e.first_name
    order by e.employee_id
)
to 'C:\Users\Public\Documents\territory_group_bridge_ETL.csv' csv delimiter ',' header;
```

*Figure 12.* Territory group extract



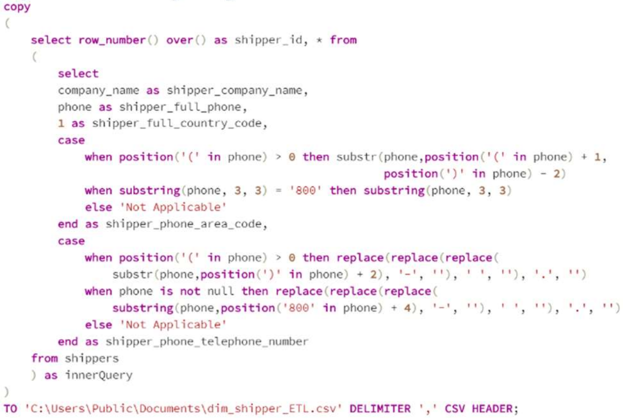
```
copy (
    select row_number() over () as territory_key,
            t.territory_id as territory_nk,
            territory_description as territory,
            region_description as region
    from employees e
    join employee_territories et
    on e.employee_id = et.employee_id
       right join territories t
    on et.territory_id = t.territory_id
        join region r
    on t.region_id = r.region_id
    order by e.employee_id
)
to 'C:\Users\Public\Documents\territory_dim_ETL.csv' csv delimiter ',' header;
```

*Figure 13.* Territory dimension extract



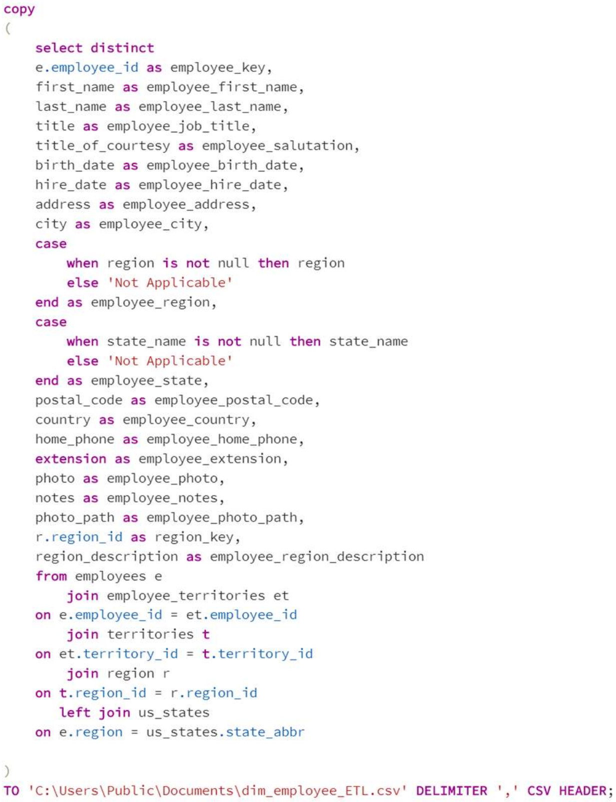
```
copy (
    select e.employee_id, row_number() over ()
    from employees e
    join employee_territories et
    on e.employee_id = et.employee_id
       join territories t
    on et.territory_id = t.territory_id
    order by e.employee_id
)
to 'C:\Users\Public\Documents\territory_bridge_ETL.csv' csv delimiter ',' header;
```

*Figure 14.* Territory bridge extract



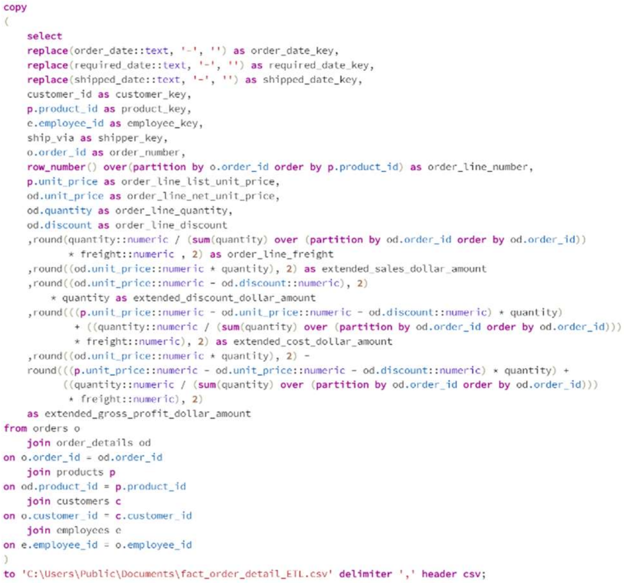
```
copy
(
    select row_number() over() as shipper_id, * from
    (
        select
        company_name as shipper_company_name,
        phone as shipper_full_phone,
        1 as shipper_full_country_code,
        case
            when position('(' in phone) > 0 then substr(phone,position('(' in phone) + 1,
                                                position(')' in phone) - 2)
            when substring(phone, 3, 3) = '800' then substring(phone, 3, 3)
            else 'Not Applicable'
        end as shipper_phone_area_code,
        case
            when position('(' in phone) > 0 then replace(replace(replace(
                substr(phone,position(')' in phone) + 2), '-', ''), ' ', ''), '.', '')
            when phone is not null then replace(replace(replace(
                substring(phone,position('800' in phone) + 4), '-', ''), ' ', ''), '.', '')
            else 'Not Applicable'
        end as shipper_phone_telephone_number
    from shippers
    ) as innerQuery
)
TO 'C:\Users\Public\Documents\dim_shipper_ETL.csv' DELIMITER ',' CSV HEADER;
```

*Figure 15.* Shipper dimension extract



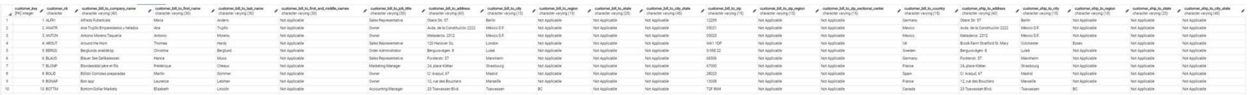
```
copy
(
    select distinct
    e.employee_id as employee_key,
    first_name as employee_first_name,
    last_name as employee_last_name,
    title as employee_job_title,
    title_of_courtesy as employee_salutation,
    birth_date as employee_birth_date,
    hire_date as employee_hire_date,
    address as employee_address,
    city as employee_city,
    case
        when region is not null then region
        else 'Not Applicable'
    end as employee_region,
    case
        when state_name is not null then state_name
        else 'Not Applicable'
    end as employee_state,
    postal_code as employee_postal_code,
    country as employee_country,
    home_phone as employee_home_phone,
    extension as employee_extension,
    photo as employee_photo,
    notes as employee_notes,
    photo_path as employee_photo_path,
    r.region_id as region_key,
    region_description as employee_region_description
    from employees e
        join employee_territories et
    on e.employee_id = et.employee_id
        join territories t
    on et.territory_id = t.territory_id
        join region r
    on t.region_id = r.region_id
       left join us_states
    on e.region = us_states.state_abbr

)
TO 'C:\Users\Public\Documents\dim_employee_ETL.csv' DELIMITER ',' CSV HEADER;
```

*Figure 16.* Employee dimension extract



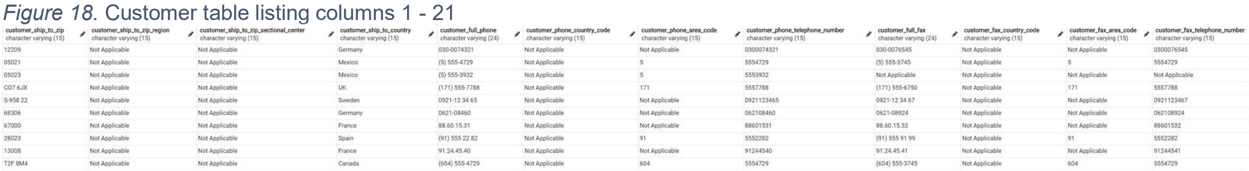
```
copy
(
    select
    replace(order_date::text, '-', '') as order_date_key,
    replace(required_date::text, '-', '') as required_date_key,
    replace(shipped_date::text, '-', '') as shipped_date_key,
    customer_id as customer_key,
    p.product_id as product_key,
    e.employee_id as employee_key,
    ship_via as shipper_key,
    o.order_id as order_number,
    row_number() over(partition by o.order_id order by p.product_id) as order_line_number,
    p.unit_price as order_line_list_unit_price,
    od.unit_price as order_line_net_unit_price,
    od.quantity as order_line_quantity,
    od.discount as order_line_discount
    ,round(quantity::numeric / (sum(quantity) over (partition by od.order_id order by od.order_id))
          * freight::numeric , 2) as order_line_freight
    ,round((od.unit_price::numeric * quantity), 2) as extended_sales_dollar_amount
    ,round((od.unit_price::numeric - od.discount::numeric), 2)
        * quantity as extended_discount_dollar_amount
    ,round(((p.unit_price::numeric - od.unit_price::numeric - od.discount::numeric) * quantity)
           + ((quantity::numeric / (sum(quantity) over (partition by od.order_id order by od.order_id)))
           * freight::numeric), 2) as extended_cost_dollar_amount
    ,round((od.unit_price::numeric * quantity), 2) -
    round(((p.unit_price::numeric - od.unit_price::numeric - od.discount::numeric) * quantity) +
          ((quantity::numeric / (sum(quantity) over (partition by od.order_id order by od.order_id))
          * freight::numeric), 2)
    as extended_gross_profit_dollar_amount
from orders o
    join order_details od
on o.order_id = od.order_id
    join products p
on od.product_id = p.product_id
    join customers c
on o.customer_id = c.customer_id
    join employees e
on e.employee_id = o.employee_id
)
to 'C:\Users\Public\Documents\fact_order_detail_ETL.csv' delimiter ',' header csv;
```

*Figure 17.* Order detail fact extract



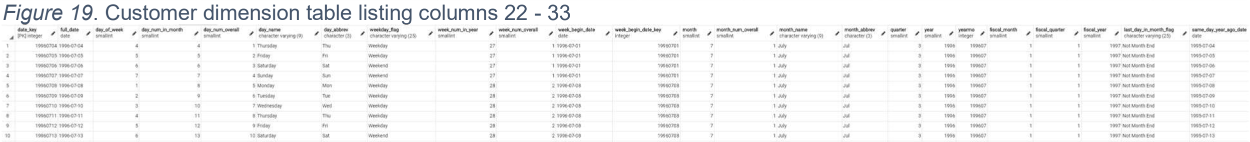
| | customer_key<br>[PK] integer | customer_nk<br>character | customer_bill_to_company_name<br>character varying (40) | customer_bill_to_first_name<br>character varying (30) | customer_bill_to_last_name<br>character varying (30) | customer_bill_to_first_and_middle_names<br>character varying (30) | customer_bill_to_job_title<br>character varying (30) | customer_bill_to_address<br>character varying (60) | customer_bill_to_city<br>character varying (15) | customer_bill_to_region<br>character varying (15) | customer_bill_to_state<br>character varying (25) | customer_bill_to_city_state<br>character varying (45) | customer_bill_to_zip<br>character varying (15) | customer_bill_to_zip_region<br>character varying (15) | customer_bill_to_zip_sectional_center<br>character varying (15) | customer_bill_to_country<br>character varying (15) | customer_ship_to_address<br>character varying (60) | customer_ship_to_city<br>character varying (15) | customer_ship_to_region<br>character varying (15) | customer_ship_to_state<br>character varying (25) | customer_ship_to_city_state<br>character varying (45) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | ALFKI | Alfreds Futterkiste | Maria | Anders | Not Applicable | Sales Representative | Obere Str. 57 | Berlin | Not Applicable | Not Applicable | Not Applicable | 12209 | Not Applicable | Not Applicable | Germany | Obere Str. 57 | Berlin | Not Applicable | Not Applicable | Not Applicable |
| 2 | 2 | ANATR | Ana Trujillo Emparedados y helados | Ana | Trujillo | Not Applicable | Owner | Avda. de la Constitución 2222 | México D.F. | Not Applicable | Not Applicable | Not Applicable | 05021 | Not Applicable | Not Applicable | Mexico | Avda. de la Constitución 2222 | México D.F. | Not Applicable | Not Applicable | Not Applicable |
| 3 | 3 | ANTON | Antonio Moreno Taquería | Antonio | Moreno | Not Applicable | Owner | Mataderos 2312 | México D.F. | Not Applicable | Not Applicable | Not Applicable | 05023 | Not Applicable | Not Applicable | Mexico | Mataderos 2312 | México D.F. | Not Applicable | Not Applicable | Not Applicable |
| 4 | 4 | AROUT | Around the Horn | Thomas | Hardy | Not Applicable | Sales Representative | 120 Hanover Sq. | London | Not Applicable | Not Applicable | Not Applicable | WA1 1DP | Not Applicable | Not Applicable | UK | Brook Farm Stratford St. Mary | Colchester | Essex | Not Applicable | Not Applicable |
| 5 | 5 | BERGS | Berglunds snabbköp | Christina | Berglund | Not Applicable | Order Administrator | Berguvsvägen 8 | Luleå | Not Applicable | Not Applicable | Not Applicable | S-958 22 | Not Applicable | Not Applicable | Sweden | Berguvsvägen 8 | Luleå | Not Applicable | Not Applicable | Not Applicable |
| 6 | 6 | BLAUS | Blauer See Delikatessen | Hanna | Moos | Not Applicable | Sales Representative | Forsterstr. 57 | Mannheim | Not Applicable | Not Applicable | Not Applicable | 68306 | Not Applicable | Not Applicable | Germany | Forsterstr. 57 | Mannheim | Not Applicable | Not Applicable | Not Applicable |
| 7 | 7 | BLONP | Blondesddsl père et fils | Frédérique | Citeaux | Not Applicable | Marketing Manager | 24, place Kléber | Strasbourg | Not Applicable | Not Applicable | Not Applicable | 67000 | Not Applicable | Not Applicable | France | 24, place Kléber | Strasbourg | Not Applicable | Not Applicable | Not Applicable |
| 8 | 8 | BOLID | Bólido Comidas preparadas | Martín | Sommer | Not Applicable | Owner | C/ Araquil, 67 | Madrid | Not Applicable | Not Applicable | Not Applicable | 28023 | Not Applicable | Not Applicable | Spain | C/ Araquil, 67 | Madrid | Not Applicable | Not Applicable | Not Applicable |
| 9 | 9 | BONAP | Bon app' | Laurence | Lebihan | Not Applicable | Owner | 12, rue des Bouchers | Marseille | Not Applicable | Not Applicable | Not Applicable | 13008 | Not Applicable | Not Applicable | France | 12, rue des Bouchers | Marseille | Not Applicable | Not Applicable | Not Applicable |
| 10 | 10 | BOTTM | Bottom-Dollar Markets | Elizabeth | Lincoln | Not Applicable | Accounting Manager | 23 Tsawassen Blvd. | Tsawassen | BC | Not Applicable | Not Applicable | T2F 8M4 | Not Applicable | Not Applicable | Canada | 23 Tsawassen Blvd. | Tsawassen | BC | Not Applicable | Not Applicable |

*Figure 18.* Customer table listing columns 1 - 21



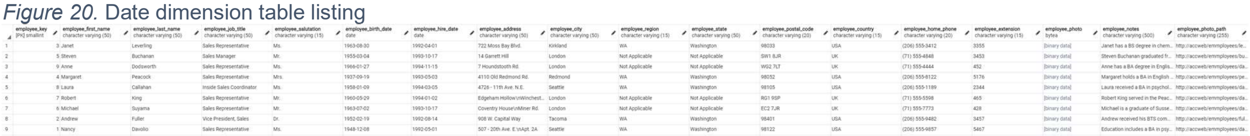
| customer_ship_to_zip<br>character varying (15) | customer_ship_to_zip_region<br>character varying (15) | customer_ship_to_zip_sectional_center<br>character varying (15) | customer_ship_to_country<br>character varying (15) | customer_full_phone<br>character varying (24) | customer_phone_country_code<br>character varying (15) | customer_phone_area_code<br>character varying (15) | customer_phone_telephone_number<br>character varying (15) | customer_full_fax<br>character varying (24) | customer_fax_country_code<br>character varying (15) | customer_fax_area_code<br>character varying (15) | customer_fax_telephone_number<br>character varying (15) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12209 | Not Applicable | Not Applicable | Germany | 030-0074321 | Not Applicable | Not Applicable | 0300074321 | 030-0076545 | Not Applicable | Not Applicable | 0300076545 |
| 05021 | Not Applicable | Not Applicable | Mexico | (5) 555-4729 | Not Applicable | 5 | 5554729 | (5) 555-3745 | Not Applicable | 5 | 5554729 |
| 05023 | Not Applicable | Not Applicable | Mexico | (5) 555-3932 | Not Applicable | 5 | 5553932 | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| CO7 6JX | Not Applicable | Not Applicable | UK | (171) 555-7788 | Not Applicable | 171 | 5557788 | (171) 555-6750 | Not Applicable | 171 | 5557788 |
| S-958 22 | Not Applicable | Not Applicable | Sweden | 0921-12 34 65 | Not Applicable | Not Applicable | 0921123465 | 0921-12 34 67 | Not Applicable | Not Applicable | 0921123467 |
| 68306 | Not Applicable | Not Applicable | Germany | 0621-08460 | Not Applicable | Not Applicable | 062108460 | 0621-08924 | Not Applicable | Not Applicable | 062108924 |
| 67000 | Not Applicable | Not Applicable | France | 88.60.15.31 | Not Applicable | Not Applicable | 88601531 | 88.60.15.32 | Not Applicable | Not Applicable | 88601532 |
| 28023 | Not Applicable | Not Applicable | Spain | (91) 555 22 82 | Not Applicable | 91 | 5552282 | (91) 555 91 99 | Not Applicable | 91 | 5552282 |
| 13008 | Not Applicable | Not Applicable | France | 91.24.45.40 | Not Applicable | Not Applicable | 91244540 | 91.24.45.41 | Not Applicable | Not Applicable | 91244541 |
| T2F 8M4 | Not Applicable | Not Applicable | Canada | (604) 555-4729 | Not Applicable | 604 | 5554729 | (604) 555-3745 | Not Applicable | 604 | 5554729 |

*Figure 19.* Customer dimension table listing columns 22 - 33



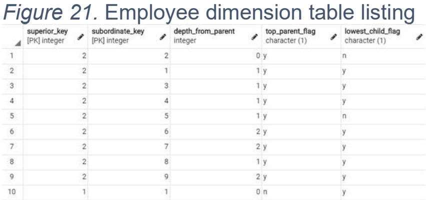
| | date_key<br>[PK] integer | full_date<br>date | day_of_week<br>smallint | day_num_in_month<br>smallint | day_num_overall<br>smallint | day_name<br>character varying (9) | day_abbrev<br>character (3) | weekday_flag<br>character varying (25) | week_num_in_year<br>smallint | week_num_overall<br>smallint | week_begin_date<br>date | week_begin_date_key<br>integer | month<br>smallint | month_num_overall<br>smallint | month_name<br>character varying (9) | month_abbrev<br>character (3) | quarter<br>smallint | year<br>smallint | yearmo<br>integer | fiscal_month<br>smallint | fiscal_quarter<br>smallint | fiscal_year<br>smallint | last_day_in_month_flag<br>character varying (25) | same_day_year_ago_date<br>date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 19960704 | 1996-07-04 | 4 | 4 | 1 | Thursday | Thu | Weekday | 27 | 1 | 1996-07-01 | 19960701 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-04 |
| 2 | 19960705 | 1996-07-05 | 5 | 5 | 2 | Friday | Fri | Weekday | 27 | 1 | 1996-07-01 | 19960701 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-05 |
| 3 | 19960706 | 1996-07-06 | 6 | 6 | 3 | Saturday | Sat | Weekend | 27 | 1 | 1996-07-01 | 19960701 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-06 |
| 4 | 19960707 | 1996-07-07 | 7 | 7 | 4 | Sunday | Sun | Weekend | 27 | 1 | 1996-07-01 | 19960701 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-07 |
| 5 | 19960708 | 1996-07-08 | 1 | 8 | 5 | Monday | Mon | Weekday | 28 | 2 | 1996-07-08 | 19960708 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-08 |
| 6 | 19960709 | 1996-07-09 | 2 | 9 | 6 | Tuesday | Tue | Weekday | 28 | 2 | 1996-07-08 | 19960708 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-09 |
| 7 | 19960710 | 1996-07-10 | 3 | 10 | 7 | Wednesday | Wed | Weekday | 28 | 2 | 1996-07-08 | 19960708 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-10 |
| 8 | 19960711 | 1996-07-11 | 4 | 11 | 8 | Thursday | Thu | Weekday | 28 | 2 | 1996-07-08 | 19960708 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-11 |
| 9 | 19960712 | 1996-07-12 | 5 | 12 | 9 | Friday | Fri | Weekday | 28 | 2 | 1996-07-08 | 19960708 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-12 |
| 10 | 19960713 | 1996-07-13 | 6 | 13 | 10 | Saturday | Sat | Weekend | 28 | 2 | 1996-07-08 | 19960708 | 7 | 1 | July | Jul | 3 | 1996 | 199607 | 1 | 1 | 1997 | Not Month End | 1995-07-13 |

*Figure 20.* Date dimension table listing



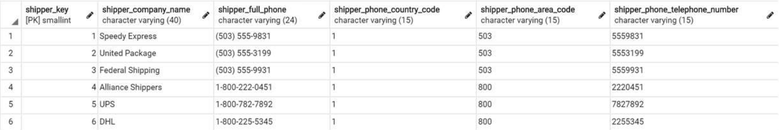
| | employee_key<br>[PK] smallint | employee_first_name<br>character varying (50) | employee_last_name<br>character varying (50) | employee_job_title<br>character varying (50) | employee_salutation<br>character varying (15) | employee_birth_date<br>date | employee_hire_date<br>date | employee_address<br>character varying (50) | employee_city<br>character varying (50) | employee_region<br>character varying (15) | employee_state<br>character varying (50) | employee_postal_code<br>character varying (20) | employee_country<br>character varying (15) | employee_home_phone<br>character varying (20) | employee_extension<br>character varying (15) | employee_photo<br>bytea | employee_notes<br>character varying (500) | employee_photo_path<br>character varying (255) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | Janet | Leverling | Sales Representative | Ms. | 1963-08-30 | 1992-04-01 | 722 Moss Bay Blvd. | Kirkland | WA | Washington | 98033 | USA | (206) 555-3412 | 3355 | [binary data] | Janet has a BS degree in chem... | http://accweb/emmployees/le... |
| 2 | 5 | Steven | Buchanan | Sales Manager | Mr. | 1955-03-04 | 1993-10-17 | 14 Garrett Hill | London | Not Applicable | Not Applicable | SW1 8JR | UK | (71) 555-4848 | 3453 | [binary data] | Steven Buchanan graduated fr... | http://accweb/emmployees/bu... |
| 3 | 9 | Anne | Dodsworth | Sales Representative | Ms. | 1966-01-27 | 1994-11-15 | 7 Houndstooth Rd. | London | Not Applicable | Not Applicable | WG2 7LT | UK | (71) 555-4444 | 452 | [binary data] | Anne has a BA degree in Englis... | http://accweb/emmployees/da... |
| 4 | 4 | Margaret | Peacock | Sales Representative | Mrs. | 1937-09-19 | 1993-05-03 | 4110 Old Redmond Rd. | Redmond | WA | Washington | 98052 | USA | (206) 555-8122 | 5176 | [binary data] | Margaret holds a BA in English ... | http://accweb/emmployees/pe... |
| 5 | 8 | Laura | Callahan | Inside Sales Coordinator | Ms. | 1958-01-09 | 1994-03-05 | 4726 - 11th Ave. N.E. | Seattle | WA | Washington | 98105 | USA | (206) 555-1189 | 2344 | [binary data] | Laura received a BA in psychol... | http://accweb/emmployees/da... |
| 6 | 7 | Robert | King | Sales Representative | Mr. | 1960-05-29 | 1994-01-02 | Edgeham Hollow\nWinchest... | London | Not Applicable | Not Applicable | RG1 9SP | UK | (71) 555-5598 | 465 | [binary data] | Robert King served in the Peac... | http://accweb/emmployees/da... |
| 7 | 6 | Michael | Suyama | Sales Representative | Mr. | 1963-07-02 | 1993-10-17 | Coventry House\nMiner Rd. | London | Not Applicable | Not Applicable | EC2 7JR | UK | (71) 555-7773 | 428 | [binary data] | Michael is a graduate of Susse... | http://accweb/emmployees/da... |
| 8 | 2 | Andrew | Fuller | Vice President, Sales | Dr. | 1952-02-19 | 1992-08-14 | 908 W. Capital Way | Tacoma | WA | Washington | 98401 | USA | (206) 555-9482 | 3457 | [binary data] | Andrew received his BTS com... | http://accweb/emmployees/ful... |
| 9 | 1 | Nancy | Davolio | Sales Representative | Ms. | 1948-12-08 | 1992-05-01 | 507 - 20th Ave. E.\nApt. 2A | Seattle | WA | Washington | 98122 | USA | (206) 555-9857 | 5467 | [binary data] | Education includes a BA in psy... | http://accweb/emmployees/da... |

*Figure 21.* Employee dimension table listing



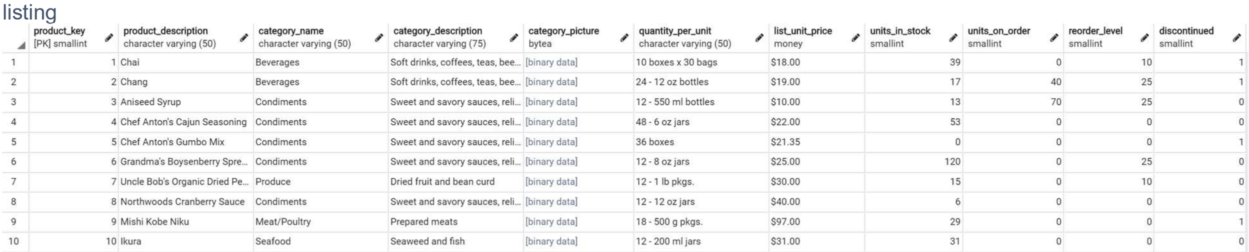
| | superior_key<br>[PK] integer | subordinate_key<br>[PK] integer | depth_from_parent<br>integer | top_parent_flag<br>character (1) | lowest_child_flag<br>character (1) |
|---|---|---|---|---|---|
| 1 | 2 | 2 | 0 | y | n |
| 2 | 2 | 1 | 1 | y | y |
| 3 | 2 | 3 | 1 | y | y |
| 4 | 2 | 4 | 1 | y | y |
| 5 | 2 | 5 | 1 | y | n |
| 6 | 2 | 6 | 2 | y | y |
| 7 | 2 | 7 | 2 | y | y |
| 8 | 2 | 8 | 1 | y | y |
| 9 | 2 | 9 | 2 | y | y |
| 10 | 1 | 1 | 0 | n | y |

*Figure 22.* Management Hierarchy Bridge table listing



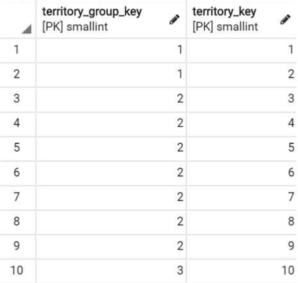
| | shipper_key<br>[PK] smallint | shipper_company_name<br>character varying (40) | shipper_full_phone<br>character varying (24) | shipper_phone_country_code<br>character varying (15) | shipper_phone_area_code<br>character varying (15) | shipper_phone_telephone_number<br>character varying (15) |
|---|---|---|---|---|---|---|
| 1 | 1 | Speedy Express | (503) 555-9831 | 1 | 503 | 5559831 |
| 2 | 2 | United Package | (503) 555-3199 | 1 | 503 | 5553199 |
| 3 | 3 | Federal Shipping | (503) 555-9931 | 1 | 503 | 5559931 |
| 4 | 4 | Alliance Shippers | 1-800-222-0451 | 1 | 800 | 2220451 |
| 5 | 5 | UPS | 1-800-782-7892 | 1 | 800 | 7827892 |
| 6 | 6 | DHL | 1-800-225-5345 | 1 | 800 | 2255345 |

*Figure 23.* Shipper dimension table listing



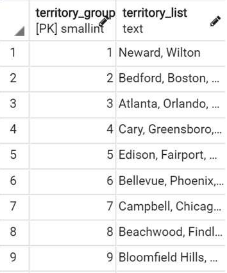
| | product_key<br>[PK] smallint | product_description<br>character varying (50) | category_name<br>character varying (50) | category_description<br>character varying (75) | category_picture<br>bytea | quantity_per_unit<br>character varying (50) | list_unit_price<br>money | units_in_stock<br>smallint | units_on_order<br>smallint | reorder_level<br>smallint | discontinued<br>smallint |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Chai | Beverages | Soft drinks, coffees, teas, bee... | [binary data] | 10 boxes x 30 bags | $18.00 | 39 | 0 | 10 | 1 |
| 2 | 2 | Chang | Beverages | Soft drinks, coffees, teas, bee... | [binary data] | 24 - 12 oz bottles | $19.00 | 17 | 40 | 25 | 1 |
| 3 | 3 | Aniseed Syrup | Condiments | Sweet and savory sauces, reli... | [binary data] | 12 - 550 ml bottles | $10.00 | 13 | 70 | 25 | 0 |
| 4 | 4 | Chef Anton's Cajun Seasoning | Condiments | Sweet and savory sauces, reli... | [binary data] | 48 - 6 oz jars | $22.00 | 53 | 0 | 0 | 0 |
| 5 | 5 | Chef Anton's Gumbo Mix | Condiments | Sweet and savory sauces, reli... | [binary data] | 36 boxes | $21.35 | 0 | 0 | 0 | 1 |
| 6 | 6 | Grandma's Boysenberry Spre... | Condiments | Sweet and savory sauces, reli... | [binary data] | 12 - 8 oz jars | $25.00 | 120 | 0 | 25 | 0 |
| 7 | 7 | Uncle Bob's Organic Dried Pe... | Produce | Dried fruit and bean curd | [binary data] | 12 - 1 lb pkgs. | $30.00 | 15 | 0 | 10 | 0 |
| 8 | 8 | Northwoods Cranberry Sauce | Condiments | Sweet and savory sauces, reli... | [binary data] | 12 - 12 oz jars | $40.00 | 6 | 0 | 0 | 0 |
| 9 | 9 | Mishi Kobe Niku | Meat/Poultry | Prepared meats | [binary data] | 18 - 500 g pkgs. | $97.00 | 29 | 0 | 0 | 1 |
| 10 | 10 | Ikura | Seafood | Seaweed and fish | [binary data] | 12 - 200 ml jars | $31.00 | 31 | 0 | 0 | 0 |

*Figure 24.* Product dimension table listing



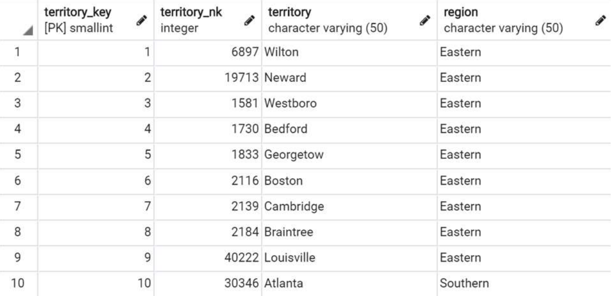
| | territory_group_key<br>[PK] smallint | territory_key<br>[PK] smallint |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 3 |
| 4 | 2 | 4 |
| 5 | 2 | 5 |
| 6 | 2 | 6 |
| 7 | 2 | 7 |
| 8 | 2 | 8 |
| 9 | 2 | 9 |
| 10 | 3 | 10 |

*Figure 25.* Territory bridge table listing



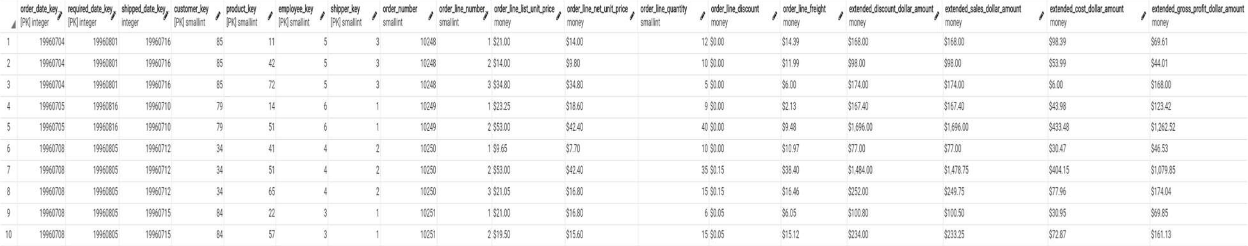
| | territory_group<br>[PK] smallint | territory_list<br>text |
|---|---|---|
| 1 | 1 | Neward, Wilton |
| 2 | 2 | Bedford, Boston, ... |
| 3 | 3 | Atlanta, Orlando, ... |
| 4 | 4 | Cary, Greensboro,... |
| 5 | 5 | Edison, Fairport, ... |
| 6 | 6 | Bellevue, Phoenix,... |
| 7 | 7 | Campbell, Chicag... |
| 8 | 8 | Beachwood, Findl... |
| 9 | 9 | Bloomfield Hills, ... |

*Figure 26.* Territory group table listing



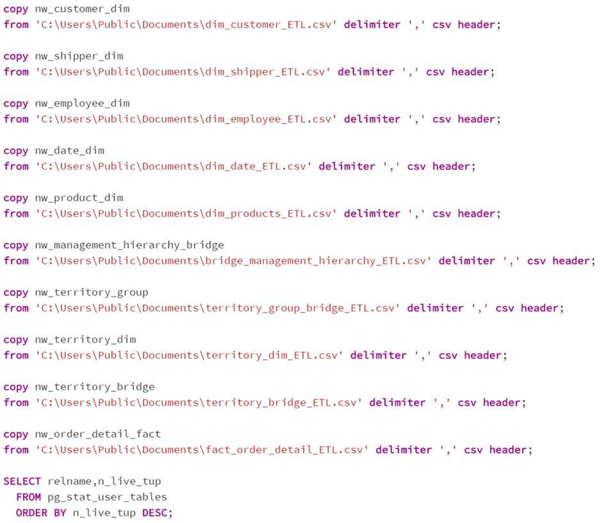
| | territory_key<br>[PK] smallint | territory_nk<br>integer | territory<br>character varying (50) | region<br>character varying (50) |
|---|---|---|---|---|
| 1 | 1 | 6897 | Wilton | Eastern |
| 2 | 2 | 19713 | Neward | Eastern |
| 3 | 3 | 1581 | Westboro | Eastern |
| 4 | 4 | 1730 | Bedford | Eastern |
| 5 | 5 | 1833 | Georgetow | Eastern |
| 6 | 6 | 2116 | Boston | Eastern |
| 7 | 7 | 2139 | Cambridge | Eastern |
| 8 | 8 | 2184 | Braintree | Eastern |
| 9 | 9 | 40222 | Louisville | Eastern |
| 10 | 10 | 30346 | Atlanta | Southern |

*Figure 27.* Territory dimension table listing



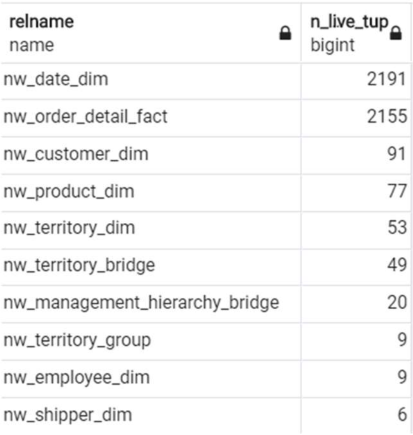
| | order_date_key<br>[PK] integer | required_date_key<br>[PK] integer | shipped_date_key<br>integer | customer_key<br>[PK] smallint | product_key<br>[PK] smallint | employee_key<br>[PK] smallint | shipper_key<br>[PK] smallint | order_number<br>smallint | order_line_number<br>smallint | order_line_list_unit_price<br>money | order_line_net_unit_price<br>money | order_line_quantity<br>smallint | order_line_discount<br>money | order_line_freight<br>money | extended_discount_dollar_amount<br>money | extended_sales_dollar_amount<br>money | extended_cost_dollar_amount<br>money | extended_gross_profit_dollar_amount<br>money |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 19960704 | 19960801 | 19960716 | 85 | 11 | 5 | 3 | 10248 | 1 | $21.00 | $14.00 | 12 | $0.00 | $14.39 | $168.00 | $168.00 | $98.39 | $69.61 |
| 2 | 19960704 | 19960801 | 19960716 | 85 | 42 | 5 | 3 | 10248 | 2 | $14.00 | $9.80 | 10 | $0.00 | $11.99 | $98.00 | $98.00 | $53.99 | $44.01 |
| 3 | 19960704 | 19960801 | 19960716 | 85 | 72 | 5 | 3 | 10248 | 3 | $34.80 | $34.80 | 5 | $0.00 | $6.00 | $174.00 | $174.00 | $6.00 | $168.00 |
| 4 | 19960705 | 19960816 | 19960710 | 79 | 14 | 6 | 1 | 10249 | 1 | $23.25 | $18.60 | 9 | $0.00 | $2.13 | $167.40 | $167.40 | $43.98 | $123.42 |
| 5 | 19960705 | 19960816 | 19960710 | 79 | 51 | 6 | 1 | 10249 | 2 | $53.00 | $42.40 | 40 | $0.00 | $9.48 | $1,696.00 | $1,696.00 | $433.48 | $1,262.52 |
| 6 | 19960708 | 19960805 | 19960712 | 34 | 41 | 4 | 2 | 10250 | 1 | $9.65 | $7.70 | 10 | $0.00 | $10.97 | $77.00 | $77.00 | $30.47 | $46.53 |
| 7 | 19960708 | 19960805 | 19960712 | 34 | 51 | 4 | 2 | 10250 | 2 | $53.00 | $42.40 | 35 | $0.15 | $38.40 | $1,484.00 | $1,478.75 | $404.15 | $1,079.85 |
| 8 | 19960708 | 19960805 | 19960712 | 34 | 65 | 4 | 2 | 10250 | 3 | $21.05 | $16.80 | 15 | $0.15 | $16.46 | $252.00 | $249.75 | $77.96 | $174.04 |
| 9 | 19960708 | 19960805 | 19960715 | 84 | 22 | 3 | 1 | 10251 | 1 | $21.00 | $16.80 | 6 | $0.05 | $6.05 | $100.80 | $100.50 | $30.95 | $69.85 |
| 10 | 19960708 | 19960805 | 19960715 | 84 | 57 | 3 | 1 | 10251 | 2 | $19.50 | $15.60 | 15 | $0.05 | $15.12 | $234.00 | $233.25 | $72.87 | $161.13 |

*Figure 28.* Order detail fact table



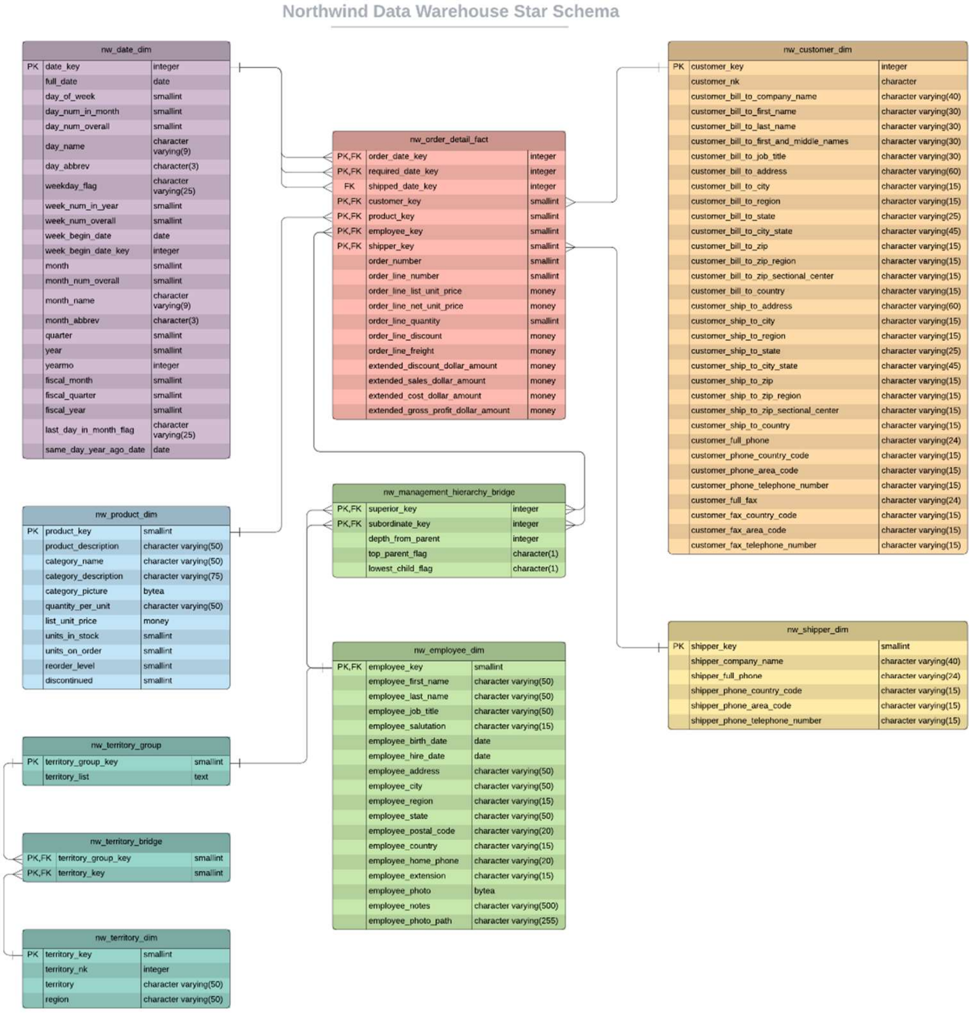
```
copy nw_customer_dim
from 'C:\Users\Public\Documents\dim_customer_ETL.csv' delimiter ',' csv header;

copy nw_shipper_dim
from 'C:\Users\Public\Documents\dim_shipper_ETL.csv' delimiter ',' csv header;

copy nw_employee_dim
from 'C:\Users\Public\Documents\dim_employee_ETL.csv' delimiter ',' csv header;

copy nw_date_dim
from 'C:\Users\Public\Documents\dim_date_ETL.csv' delimiter ',' csv header;

copy nw_product_dim
from 'C:\Users\Public\Documents\dim_products_ETL.csv' delimiter ',' csv header;

copy nw_management_hierarchy_bridge
from 'C:\Users\Public\Documents\bridge_management_hierarchy_ETL.csv' delimiter ',' csv header;

copy nw_territory_group
from 'C:\Users\Public\Documents\territory_group_bridge_ETL.csv' delimiter ',' csv header;

copy nw_territory_dim
from 'C:\Users\Public\Documents\territory_dim_ETL.csv' delimiter ',' csv header;

copy nw_territory_bridge
from 'C:\Users\Public\Documents\territory_bridge_ETL.csv' delimiter ',' csv header;

copy nw_order_detail_fact
from 'C:\Users\Public\Documents\fact_order_detail_ETL.csv' delimiter ',' csv header;

SELECT relname,n_live_tup
  FROM pg_stat_user_tables
  ORDER BY n_live_tup DESC;
```

*Figure 29.* Load statements and SQL to generate table counts



| relname<br>name | n_live_tup<br>bigint |
|---|---|
| nw_date_dim | 2191 |
| nw_order_detail_fact | 2155 |
| nw_customer_dim | 91 |
| nw_product_dim | 77 |
| nw_territory_dim | 53 |
| nw_territory_bridge | 49 |
| nw_management_hierarchy_bridge | 20 |
| nw_territory_group | 9 |
| nw_employee_dim | 9 |
| nw_shipper_dim | 6 |

*Figure 30.* Row counts of all ten tables



| nw_date_dim | | |
|---|---|---|
| PK | date_key | integer |
| | full_date | date |
| | day_of_week | smallint |
| | day_num_in_month | smallint |
| | day_num_overall | smallint |
| | day_name | character varying(9) |
| | day_abbrev | character(3) |
| | weekday_flag | character varying(25) |
| | week_num_in_year | smallint |
| | week_num_overall | smallint |
| | week_begin_date | date |
| | week_begin_date_key | integer |
| | month | smallint |
| | month_num_overall | smallint |
| | month_name | character varying(9) |
| | month_abbrev | character(3) |
| | quarter | smallint |
| | year | smallint |
| | yearmo | integer |
| | fiscal_month | smallint |
| | fiscal_quarter | smallint |
| | fiscal_year | smallint |
| | last_day_in_month_flag | character varying(25) |
| | same_day_year_ago_date | date |

| nw_order_detail_fact | | |
|---|---|---|
| PK,FK | order_date_key | integer |
| PK,FK | required_date_key | integer |
| FK | shipped_date_key | integer |
| PK,FK | customer_key | smallint |
| PK,FK | product_key | smallint |
| PK,FK | employee_key | smallint |
| PK,FK | shipper_key | smallint |
| | order_number | smallint |
| | order_line_number | smallint |
| | order_line_list_unit_price | money |
| | order_line_net_unit_price | money |
| | order_line_quantity | smallint |
| | order_line_discount | money |
| | order_line_freight | money |
| | extended_discount_dollar_amount | money |
| | extended_sales_dollar_amount | money |
| | extended_cost_dollar_amount | money |
| | extended_gross_profit_dollar_amount | money |

| nw_customer_dim | | |
|---|---|---|
| PK | customer_key | integer |
| | customer_nk | character |
| | customer_bill_to_company_name | character varying(40) |
| | customer_bill_to_first_name | character varying(30) |
| | customer_bill_to_last_name | character varying(30) |
| | customer_bill_to_first_and_middle_names | character varying(30) |
| | customer_bill_to_job_title | character varying(30) |
| | customer_bill_to_address | character varying(60) |
| | customer_bill_to_city | character varying(15) |
| | customer_bill_to_region | character varying(15) |
| | customer_bill_to_state | character varying(25) |
| | customer_bill_to_city_state | character varying(45) |
| | customer_bill_to_zip | character varying(15) |
| | customer_bill_to_zip_region | character varying(15) |
| | customer_bill_to_zip_sectional_center | character varying(15) |
| | customer_bill_to_country | character varying(15) |
| | customer_ship_to_address | character varying(60) |
| | customer_ship_to_city | character varying(15) |
| | customer_ship_to_region | character varying(15) |
| | customer_ship_to_state | character varying(25) |
| | customer_ship_to_city_state | character varying(45) |
| | customer_ship_to_zip | character varying(15) |
| | customer_ship_to_zip_region | character varying(15) |
| | customer_ship_to_zip_sectional_center | character varying(15) |
| | customer_ship_to_country | character varying(15) |
| | customer_full_phone | character varying(24) |
| | customer_phone_country_code | character varying(15) |
| | customer_phone_area_code | character varying(15) |
| | customer_phone_telephone_number | character varying(15) |
| | customer_full_fax | character varying(24) |
| | customer_fax_country_code | character varying(15) |
| | customer_fax_area_code | character varying(15) |
| | customer_fax_telephone_number | character varying(15) |

| nw_product_dim | | |
|---|---|---|
| PK | product_key | smallint |
| | product_description | character varying(50) |
| | category_name | character varying(50) |
| | category_description | character varying(75) |
| | category_picture | bytea |
| | quantity_per_unit | character varying(50) |
| | list_unit_price | money |
| | units_in_stock | smallint |
| | units_on_order | smallint |
| | reorder_level | smallint |
| | discontinued | smallint |

| nw_management_hierarchy_bridge | | |
|---|---|---|
| PK,FK | superior_key | integer |
| PK,FK | subordinate_key | integer |
| | depth_from_parent | integer |
| | top_parent_flag | character(1) |
| | lowest_child_flag | character(1) |

| nw_shipper_dim | | |
|---|---|---|
| PK | shipper_key | smallint |
| | shipper_company_name | character varying(40) |
| | shipper_full_phone | character varying(24) |
| | shipper_phone_country_code | character varying(15) |
| | shipper_phone_area_code | character varying(15) |
| | shipper_phone_telephone_number | character varying(15) |

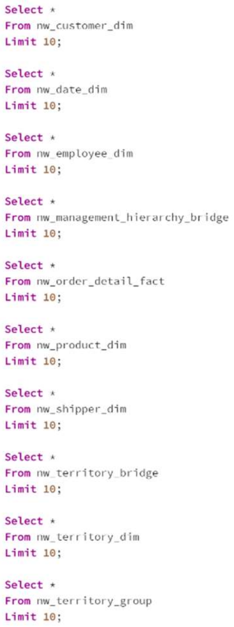
| nw_employee_dim | | |
|---|---|---|
| PK,FK | employee_key | smallint |
| | employee_first_name | character varying(50) |
| | employee_last_name | character varying(50) |
| | employee_job_title | character varying(50) |
| | employee_salutation | character varying(15) |
| | employee_birth_date | date |
| | employee_hire_date | date |
| | employee_address | character varying(50) |
| | employee_city | character varying(50) |
| | employee_region | character varying(15) |
| | employee_state | character varying(50) |
| | employee_postal_code | character varying(20) |
| | employee_country | character varying(15) |
| | employee_home_phone | character varying(20) |
| | employee_extension | character varying(15) |
| | employee_photo | bytea |
| | employee_notes | character varying(500) |
| | employee_photo_path | character varying(255) |

| nw_territory_group | | |
|---|---|---|
| PK | territory_group_key | smallint |
| | territory_list | text |

| nw_territory_bridge | | |
|---|---|---|
| PK,FK | territory_group_key | smallint |
| PK,FK | territory_key | smallint |

| nw_territory_dim | | |
|---|---|---|
| PK | territory_key | smallint |
| | territory_nk | integer |
| | territory | character varying(50) |
| | region | character varying(50) |

*Figure 31.* Northwind Data Warehouse Star Schema



```
Select *
From nw_customer_dim
Limit 10;

Select *
From nw_date_dim
Limit 10;

Select *
From nw_employee_dim
Limit 10;

Select *
From nw_management_hierarchy_bridge
Limit 10;

Select *
From nw_order_detail_fact
Limit 10;

Select *
From nw_product_dim
Limit 10;

Select *
From nw_shipper_dim
Limit 10;

Select *
From nw_territory_bridge
Limit 10;

Select *
From nw_territory_dim
Limit 10;

Select *
From nw_territory_group
Limit 10;
```

*Figure 32.* SQL to create table listings

# References

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling (Third edition).* John Wiley & Sons, Inc.

McHugh, J. (2017, March 8). *Data Warehouse Design Techniques—Ragged Hierarchical Dimensions.* https://www.nuwavesolutions.com/ragged_hierarchical_dimensions/

McHugh, J. (2017, May 24). Loading Hierarchical Bridge Tables. https://www.nuwavesolutions.com/loading-hierarchical-bridge-tables/