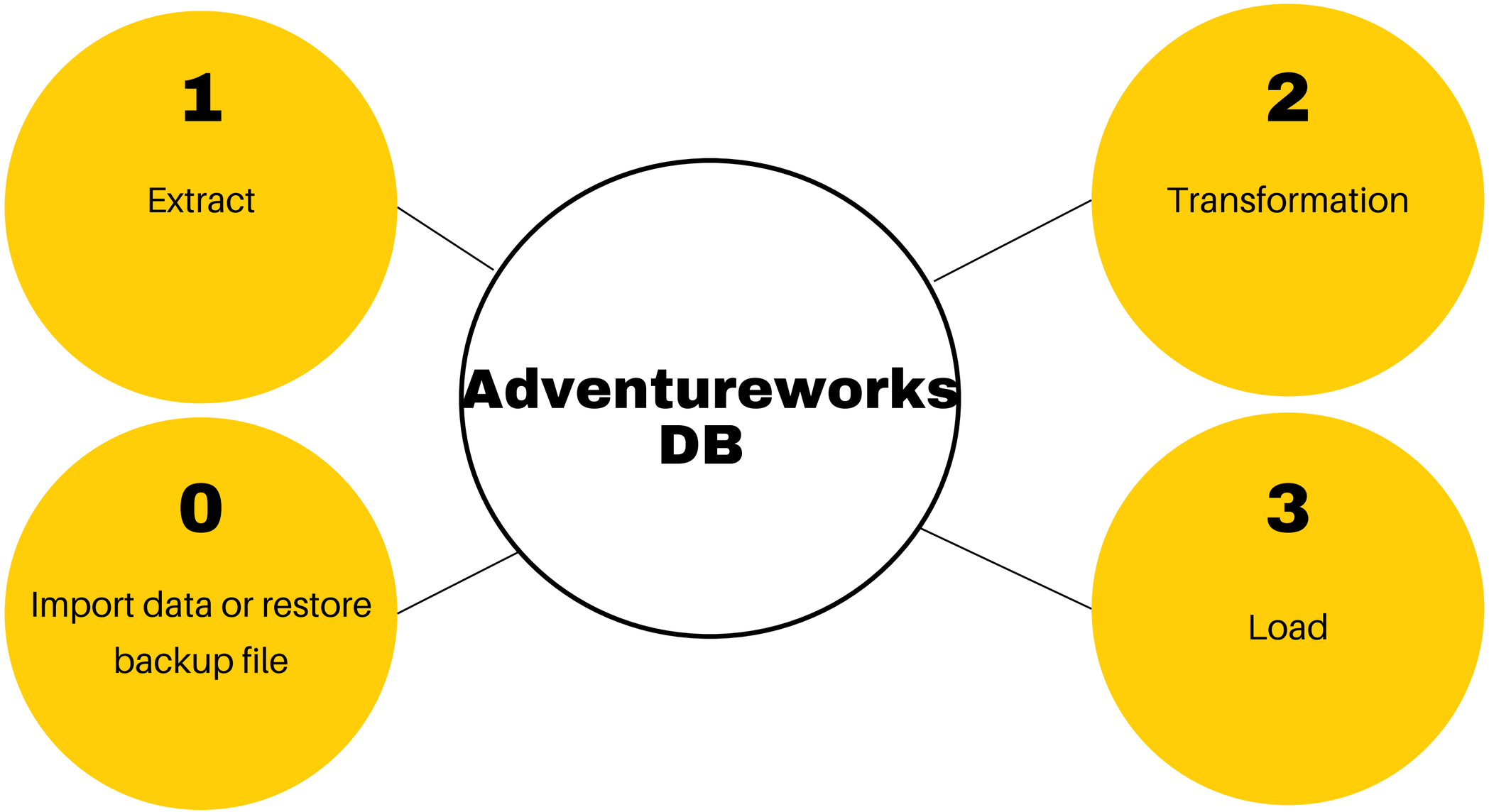
ETL

Table of Content

- ADVENTUREWORKS DATABASE
- IMPORT - BACKUP RESTORE
- ETL OPERATIONS
- DEMONSTRATION
- Q&A


1
Extract
2
Transformation
Adventureworks DB
0
Import data or restore backup file
3
Load

ETL- POSTGRESQL

Set up sample database for PostgreSQL

Download the backup file and restore it in your database. Backup file Link: https://github.com/daniuae/training/commit/b03e11c62146c1c06346f7d2b90a1c74ae50d8f2

Pre-requsites

Ensure you have:

- PostgreSQL installed and running.
- psql or PgAdmin access.
- The AdventureWorks database loaded into PostgreSQL.

Extract Data (from source tables)
Extract data from transactional tables like sales.salesorderheader and sales.salesorderdetail.

Transform Data:
Perform basic transformations like:

- Date formatting
- Calculating revenue
- Joining with dimension tables (e.g., product names)

Our Service

BRANDING

Lorem ipsum dolor sit a eleifend, mis finibus turpis, a lacinia est tellus sit amet turpis.

DIGITAL MARKETING

Lorem ipsum dolor sit a eleifend, mis finibus turpis, a lacinia est tellus sit amet turpis.

BUSINESS DEVELOPMENT

Lorem ipsum dolor sit a eleifend, mis finibus turpis, a lacinia est tellus sit amet turpis.

MARKETING PLAN

Lorem ipsum dolor sit a eleifend, mis finibus turpis, a lacinia est tellus sit amet turpis.

EXTRACT

```
-- Extract raw sales data (join header and details)
CREATE TABLE etl_stage_sales AS
SELECT
    soh.salesorderid,
    soh.orderdate,
    soh.customerid,
    sod.productid,
    sod.orderqty,
    sod.unitprice,
    sod.unitprice * sod.orderqty AS total_price
FROM sales.salesorderheader soh
JOIN sales.salesorderdetail sod
  ON soh.salesorderid = sod.salesorderid;
```

TRANSFORM

```
-- Add product and customer info
CREATE TABLE etl_transformed_sales AS
SELECT
    s.salesorderid,
    TO_CHAR(s.orderdate, 'YYYY-MM-DD') AS order_date,
    --c.firstname || ' ' || c.lastname AS customer_name,
   c.customerid AS customer_name,
    p.name AS product_name,
    s.orderqty,
    s.unitprice,
    s.total_price
FROM etl_stage_sales s
LEFT JOIN sales.customer c ON s.customerid = c.customerid
LEFT JOIN production.product p ON s.productid = p.productid;
```

LOAD

```
CREATE TABLE report_sales_summary (
    salesorderid INT,
    order_date DATE,
    customer_name TEXT,
    product_name TEXT,
    orderqty INT,
    unitprice NUMERIC,
    total_price NUMERIC
);
-- Load transformed data
INSERT INTO report_sales_summary (
    salesorderid, order_date, customer_name,
    product_name, orderqty, unitprice, total_price
)
SELECT
    salesorderid,
    order_date::DATE,
    customer_name,
    product_name,
    orderqty,
    unitprice,
    total_price
FROM etl_transformed_sales;
```


```
CREATE TABLE report_sales_summary (
    salesorderid INT,
    order_date DATE,
    customer_name TEXT,
    product_name TEXT,
    orderqty INT,
    unitprice NUMERIC,
    total_price NUMERIC
);
-- Load transformed data
INSERT INTO report_sales_summary (
    salesorderid, order_date, customer_name,
    product_name, orderqty, unitprice, total_price
)
SELECT
    salesorderid,
    order_date::DATE,
    customer_name,
    product_name,
    orderqty,
    unitprice,
    total_price
FROM etl_transformed_sales;

-- Row count
SELECT COUNT(*) FROM report_sales_summary;

-- Total sales
SELECT SUM(total_price) FROM report_sales_summary;

-- Sales by product
SELECT product_name, SUM(total_price) AS revenue
FROM report_sales_summary
GROUP BY product_name
ORDER BY revenue DESC;
```

Q & A