E-Commerce Project Documentation

Project Overview

Project Overview:

Ecommerce website dataset is ready for you to build data warehouse for to answer business questions. The project consist of 3 main phases.

1. Building data warehouse model to serve business queries.
2. Building ETL pipeline to land data into the warehouse.
3. Run analytical queries to answer business questions

Resources:

Ecommerce data set on git project folder

Requirements:

- Design Data warehouse schema.
- Build ETL pipeline to load data into DW
- Answer Below business questions:
    1. When is the peak season of our ecommerce?
    2. What time users are most likely make an order or using the ecommerce app?
    3. What is the preferred way to pay in the ecommerce?
    4. How many installments is usually done when paying in the ecommerce?
    5. What is the average spending time for user for our ecommerce?
    6. What is the frequency of purchase on each state?
    7. Which logistic route that have heavy traffic in our ecommerce?

8. How many late delivered order in our ecommerce? Are late order affecting the customer satisfaction?
9. How long are the delay for delivery / shipping process in each state?
10. How long are the difference between estimated delivery time and actual delivery time in each state?

Understanding the Data

The data consists of 7 CSV files:

- `order_item_dataset`: **contains the price and shipping cost for every item in every order made by customers.**
- `order_dataset`: **Details about orders such as order status and dates of different order statuses such as order date, order approved date, pickup date, delivery date, estimated time delivery.**
- `seller_dataset`: **Data about different sellers and their locations.**
- `user_dataset`: **Data about different customers and their locations. Some users exist in the dataset more than once due to changes in their location information.**
- `product_dataset`: **All available products in the e-commerce platform.**
- `feedback_dataset`: **Feedback scores for each order and the dates when feedback forms were sent to and answered by customers. Some orders have multiple feedback forms associated with them.**
- `payment_dataset`: **contains payment information for each payment done for each order.**

Each order has the following stages:

1. **Created:** The order is placed in the system by the customer.
2. **Approved:** The order is reviewed and approved, often after verifying payment and stock availability.
3. **Invoiced:** An invoice is generated for the order, confirming the details and payment.

4. **Processing:** The order is being prepared, such as packaging the product or allocating stock.
5. **Shipped:** The order has left the warehouse or fulfillment center and is in transit to the customer.
6. **Delivered:** The order has reached the customer successfully, completing the transaction.

During any stage of this process a customer can cancel the order. If the product is out of stock or cannot be fulfilled, the order status will be "Unavailable"

The platform supports the following payment methods:

1. **Voucher**: A pre-paid code or ticket that can be exchanged for goods, services, or discounts. It's like a gift card with a specific value or purpose.
2. **Credit Card**: A card that lets you borrow money from a bank to make purchases. You pay it back later, often with interest if not paid in full by the due date.
3. **Debit Card**: A card linked to your bank account that directly uses your own money to pay for purchases. It deducts the amount immediately from your account.
4. **Blipay**: A digital payment method (often used in specific regions or platforms) that allows you to make transactions electronically, typically through a mobile app or online system. It's like a digital wallet.

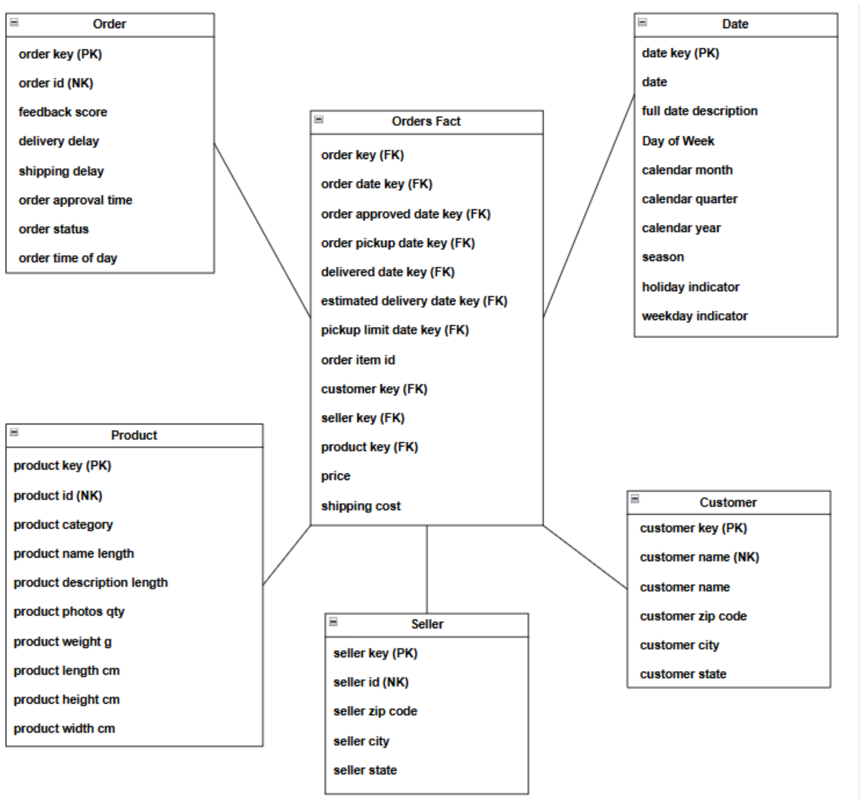
The platform allows customers to split their payments into multiple transactions or methods. For example, a customer could pay part of the amount with a credit card and the rest with a voucher or a different payment method. The `payment_sequential` column in the `payment_dataset` is used to track the sequence of these payments for a single order.

Data Modelling

Based on my understanding of the data, there's a business process which is **ordering** with dimensions: **order, customer, seller, product, and Date.** And a granularity of one row for each item in an order. The measurements for this business process are **price** and **shipping cost.** And since payment data isn't at the same grain as ordering data, payment information is captured in a separate fact table with granularity of one row per payment per order, and dimensions **order and payment method.**

Ordering

The below figure shows the data model for the ordering business process:



Date Dimension

The dataset includes multiple dates, and business users often need to filter or group data based on date-related attributes such as month, quarter, or year. For example, one of the key business questions is identifying the platform's peak season. To support this analysis, a **Date Dimension** is included in the ordering business process. This dimension spans 20 years, starting from 2016, and contains individual rows for each day within that period.

Role Playing

As mentioned earlier, multiple dates are associated with an order (e.g., order date, delivery date, etc.). Each date associated with an order is linked to the Date Dimension through a foreign key in the fact table.

Missing Dates

Some dates may not be applicable for specific order stages, for example, if the order hasn’t been delivered yet then it doesn’t make sense to have a value for the delivery date. So, for some orders there’s a possibility of having missing dates. To prevent null foreign keys in the fact table, a special “unknown” record is added to the date dimension so that missing dates reference that record, maintaining referential integrity and ensuring consistency in the data model.

Order Dimension

- The **Order Dimension** is designed to store descriptive attributes related to each order, such as **order status** and **order time of day**. The **order time of day** is derived from the time component of the order date timestamp, providing a clear breakdown of when orders are placed throughout the day.
- The **feedback score**, although a numeric field, is included in the Order Dimension rather than the fact table. This is because the feedback score represents a discrete set of values rather than a continuous range. The feedback score is sourced from the feedback dataset, where some orders may have multiple feedback forms associated with them. To maintain consistency, only the first feedback score linked to each order is used in this dimension.
- Additionally, metrics such as **shipping delay**, **delivery delay**, and **order approval time** are also stored in the Order Dimension. While these attributes are numeric, they are stored in the order dimension table and not the fact table because they don’t align with the fact table’s granularity of one row per order item. Including these metrics in the Order Dimension ensures efficient processing and avoids potential confusion by maintaining a consistent and unified method of calculation. These values are computed using date-related data associated with each order, further enhancing the clarity and usability of the dimension.

Customer

The **Customer** dimension is derived from the users dataset. This dataset contained multiple rows for some customers, with each row representing different address information. However, the dataset did not specify which address was the current one or provide valid time periods for each address. The decision was made to retain only one row per customer, discarding duplicate entries

Seller and Product

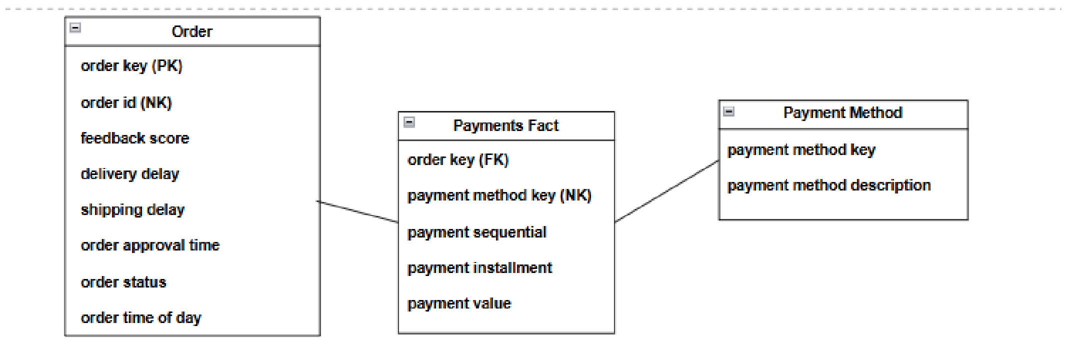
The **Seller** and **Product** dimensions were sourced directly from the seller and product datasets, respectively. These datasets required no transformations, as they were already structured in a way that aligned with the dimensional model.

Orders Fact

The **Orders Fact** table is designed with a granularity of one row per order item. This level of granularity was chosen because it represents the most atomic level of data, enabling business users to analyze facts at both the order level and the individual order item level. The table's primary key is a composite key, consisting of the **order key** (a foreign key) and the **order item ID** (a degenerate dimension). Additionally, the table includes foreign keys linking to the **order dates**, **order**, **product**, **seller**, and **customer** dimensions. The key facts captured in this table are the **item price** and **item shipping cost**, providing essential metrics for analysis and reporting.

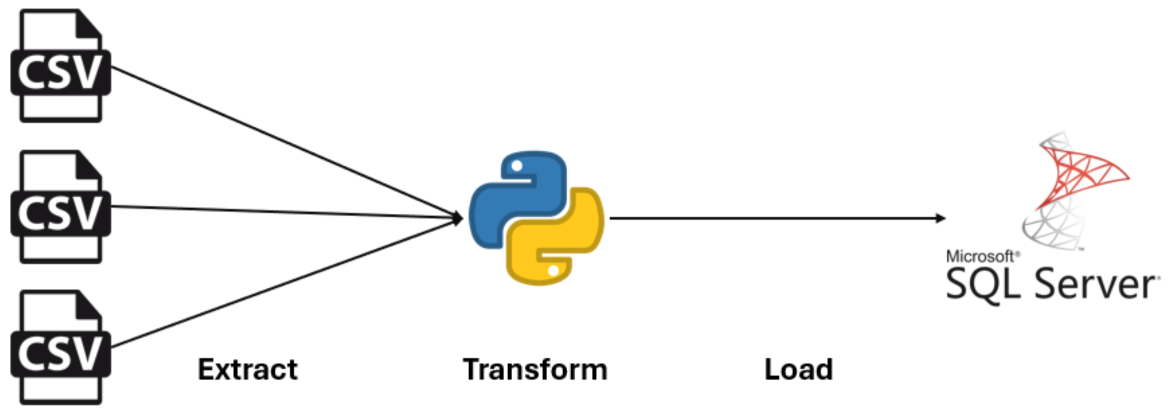
Payment

The below figure shows the data model for the payment business process:



The order dimension in this process aligns directly with the dimension used in the ordering business process. The payment method dimension provides detailed descriptions for each payment method supported by the platform. The payments fact table is designed with a granularity of one payment transaction per order. Its primary key is a composite key, comprising foreign keys for the order key and payment method key, along with a payment sequential identifier. This table captures key metrics, including the number of payment installments and the payment value.

ETL Pipeline



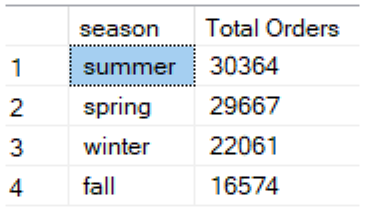
- Python was used as an ETL tool to extract the data from the given CSV file, perform necessary transformations, and load the data into the data model.
- ETL scripts for the order data model can be found in the `Orders` directory, and scripts for payment data model can be found in the `Payment` directory.
- The file `main.py` is the entry point for running the pipeline to load both data models.

Business Questions Answered

When is the peak season of our ecommerce?

```
SELECT season, COUNT(DISTINCT order_key) AS [Total Orders]
FROM Orders_Fact f
JOIN Date d
ON f.order_date_key = d.date_key
GROUP BY season
ORDER BY [Total Orders] DESC
```

Result:

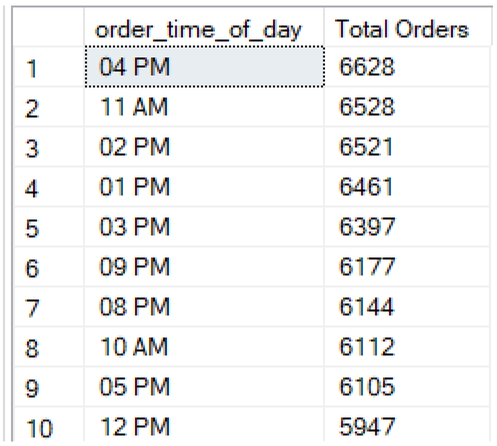


	season	Total Orders
1	summer	30364
2	spring	29667
3	winter	22061
4	fall	16574

What time users are most likely make an order or using the ecommerce app?

```
SELECT order_time_of_day, COUNT(DISTINCT f.order_key) AS [Tot
al Orders]
FROM Orders_Fact f
JOIN [Order] o
ON f.order_key = o.order_key
GROUP BY order_time_of_day
ORDER BY [Total Orders] DESC
```

Result:

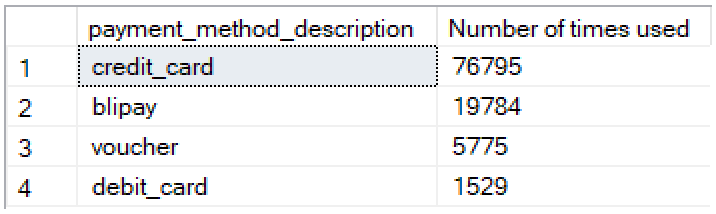


	order_time_of_day	Total Orders
1	04 PM	6628
2	11 AM	6528
3	02 PM	6521
4	01 PM	6461
5	03 PM	6397
6	09 PM	6177
7	08 PM	6144
8	10 AM	6112
9	05 PM	6105
10	12 PM	5947

What is the preferred way to pay in the ecommerce?

```
SELECT
    payment_method_description,
    COUNT(payment_method_description) AS [Number of times use
d]
FROM
    payment_fact pf
JOIN
    payment_method pm
ON
    pf.payment_method_key = pm.payment_method_key
WHERE
    payment_method_description != 'not_defined'GROUP BY
    payment_method_description
ORDER BY
    [Number of times used] DESC
```

Result:

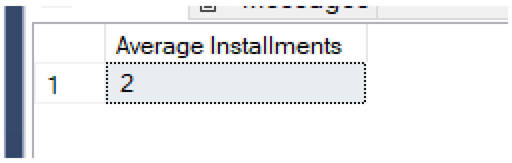


	payment_method_description	Number of times used
1	credit_card	76795
2	blipay	19784
3	voucher	5775
4	debit_card	1529

How many installment is usually done when paying in the ecommerce?

```
SELECT
    AVG(payment_installments) AS [Average Installments]
FROM
    payment_fact pf
JOIN
    payment_method pm
ON
    pf.payment_method_key = pm.payment_method_key
```

Result:



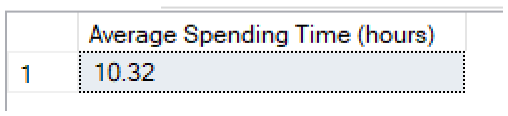
	Average Installments
1	2

What is the average spending time for user for our ecommerce?

Based on my understanding of the data, the spending time for a user is the duration from when the user places the order until the payment is approved, which is the difference between `order_date` and `order_approval_date`

```
SELECT
  ROUND(AVG(order_approval_time) / 60, 2) AS [Average Spendin
g Time (hours) ]
FROM(
  SELECT
      DISTINCT o.order_key, order_approval_time
  FROM
      Orders_Fact f
  JOIN
      [Order] o
  ON
      f.order_key = o.order_key
) S
```

Result:

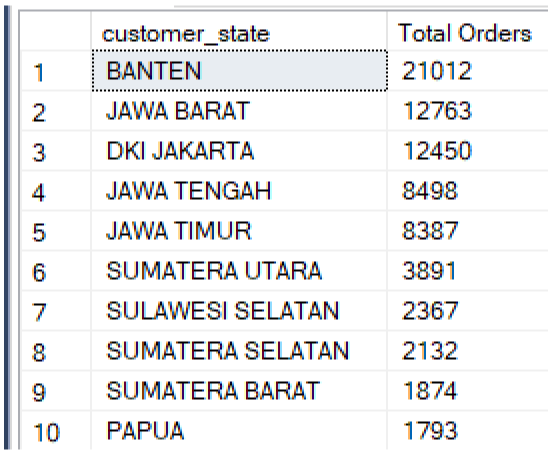


	Average Spending Time (hours)
1	10.32

What is the frequency of purchase on each state?

```
SELECT
  customer_state,
  COUNT(DISTINCT order_key) AS [Total Orders]
FROM
  Orders_Fact f
JOIN
  Customer c
ON
  f.customer_key = c.customer_key
GROUP BY
  customer_state
ORDER BY
  [Total Orders] DESC
```

Result:

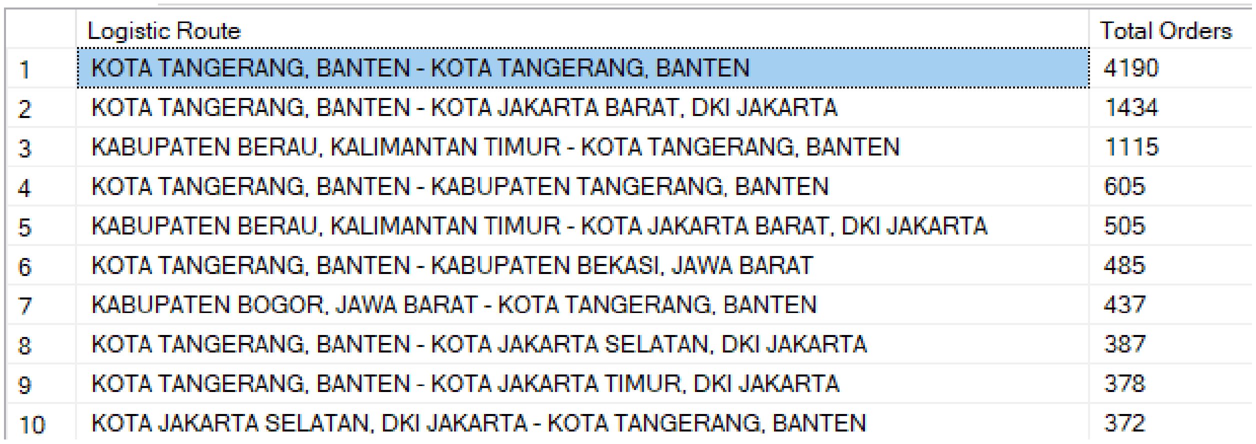


	customer_state	Total Orders
1	BANTEN	21012
2	JAWA BARAT	12763
3	DKI JAKARTA	12450
4	JAWA TENGAH	8498
5	JAWA TIMUR	8387
6	SUMATERA UTARA	3891
7	SULAWESI SELATAN	2367
8	SUMATERA SELATAN	2132
9	SUMATERA BARAT	1874
10	PAPUA	1793

Which logistic route that have heavy traffic in our ecommerce?

```
SELECT
  CONCAT(seller_city, ', ', seller_state, ' - ', customer_cit
y, ', ', customer_state) AS [Logistic Route],
  COUNT(DISTINCT order_key) AS [Total Orders]
FROM
  Orders_Fact f
JOIN
  Seller s
ON
  f.seller_key = s.seller_key
JOIN
  Customer c
ON
  f.customer_key = c.customer_key
GROUP BY
  CONCAT(seller_city, ', ', seller_state, ' - ', customer_cit
y, ', ', customer_state)
ORDER BY
  [Total Orders] DESC
```

Result:



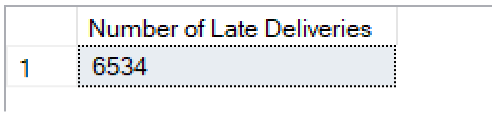
	Logistic Route	Total Orders
1	KOTA TANGERANG, BANTEN - KOTA TANGERANG, BANTEN	4190
2	KOTA TANGERANG, BANTEN - KOTA JAKARTA BARAT, DKI JAKARTA	1434
3	KABUPATEN BERAU, KALIMANTAN TIMUR - KOTA TANGERANG, BANTEN	1115
4	KOTA TANGERANG, BANTEN - KABUPATEN TANGERANG, BANTEN	605
5	KABUPATEN BERAU, KALIMANTAN TIMUR - KOTA JAKARTA BARAT, DKI JAKARTA	505
6	KOTA TANGERANG, BANTEN - KABUPATEN BEKASI, JAWA BARAT	485
7	KABUPATEN BOGOR, JAWA BARAT - KOTA TANGERANG, BANTEN	437
8	KOTA TANGERANG, BANTEN - KOTA JAKARTA SELATAN, DKI JAKARTA	387
9	KOTA TANGERANG, BANTEN - KOTA JAKARTA TIMUR, DKI JAKARTA	378
10	KOTA JAKARTA SELATAN, DKI JAKARTA - KOTA TANGERANG, BANTEN	372

How many late delivered order in our ecommerce? Are late order affecting the customer satisfaction?

Number of late deliveries:

```
SELECT
    COUNT(DISTINCT f.order_key) AS [Number of Late Deliverie
s]
FROM
    Orders_Fact f
JOIN
    [Order] o
ON
    f.order_key = o.order_key
WHERE
    order_status = 'delivered' AND delivery_delay > 0
```

Result:

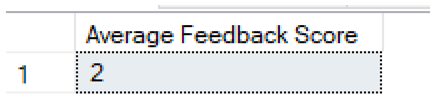


	Number of Late Deliveries
1	6534

Average feedback score for late orders:

```
WITH late_orders AS (
  SELECT DISTINCT f.order_key, feedback_score
  FROM Orders_Fact f
  JOIN [Order] o
  ON f.order_key = o.order_key
  WHERE order_status = 'delivered' AND delivery_delay > 0
)
SELECT AVG(feedback_score) AS [Average Feedback Score]
FROM late_orders
```

Result:



	Average Feedback Score
1	2

The average feedback score is low for late orders, so late deliveries are affecting customer satisfaction.

How long are the delay for delivery / shipping process in each state?

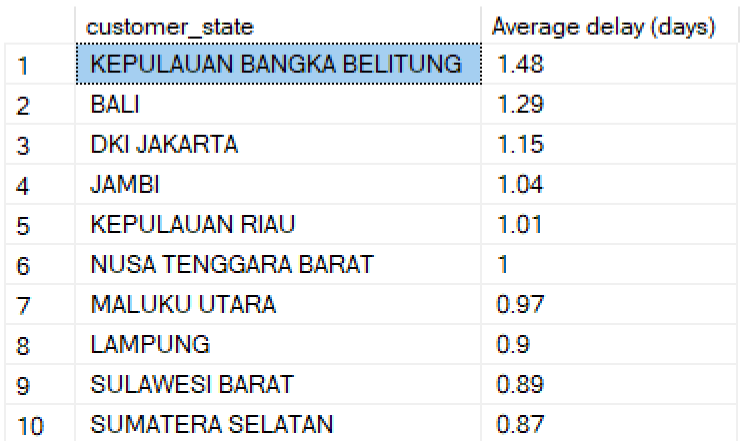
Delay for delivery can be defined as either:

- The number of days between delivery date and estimated time delivery. In this case the answer will be:

```
SELECT
  customer_state,
  ROUND(AVG(CAST(delivery_delay AS FLOAT)),2) AS [Average del
ay (days)]
FROM(
  SELECT
      DISTINCT f.order_key,
      delivery_delay ,
      customer_state
  FROM
```

```
        Orders_Fact f
    JOIN
        [Order] o
    ON
        f.order_key = o.order_key
    JOIN
        Customer c
    ON
        f.customer_key = c.customer_key
) S
GROUP BY
    customer_state
ORDER BY
    [Average delay (days)] DESC
```

Result:

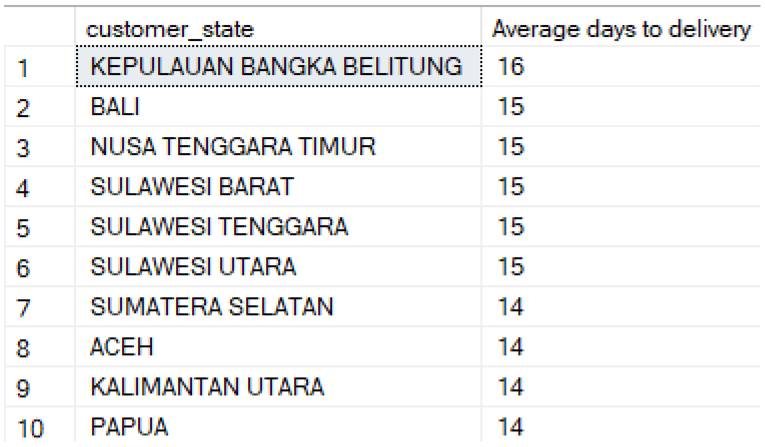


	customer_state	Average delay (days)
1	KEPULAUAN BANGKA BELITUNG	1.48
2	BALI	1.29
3	DKI JAKARTA	1.15
4	JAMBI	1.04
5	KEPULAUAN RIAU	1.01
6	NUSA TENGGARA BARAT	1
7	MALUKU UTARA	0.97
8	LAMPUNG	0.9
9	SULAWESI BARAT	0.89
10	SUMATERA SELATAN	0.87

- Or it can be the number of days between order date and delivery date, and the answer in this case will be:

```
WITH order_dates AS (
    SELECT
        DISTINCT order_key,
        customer_state,
        CAST(d1.date AS DATE) AS order_date,
        CAST(d2.date AS DATE) AS delivery_date
    FROM
        Orders_Fact f
    JOIN Date d1
        ON d1.date_key = f.order_date_key
    JOIN Date d2
        ON d2.date_key = f.delivered_date_key
    JOIN Customer c
        ON c.customer_key = f.customer_key
    WHERE d1.date != 'unknown' AND d2.date != 'unknown'
)
SELECT
    customer_state,
    AVG(DATEDIFF(DAY,order_date, delivery_date)) AS [Average
days to delivery]
FROM
    order_dates
GROUP BY
    customer_state
ORDER BY
    [Average days to delivery] DESC
```

Result:



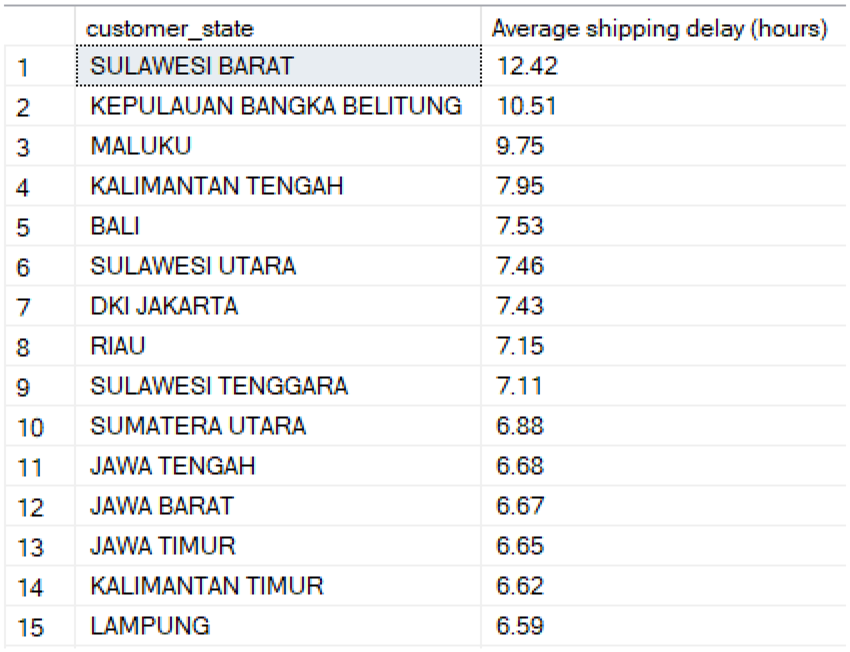
	customer_state	Average days to delivery
1	KEPULAUAN BANGKA BELITUNG	16
2	BALI	15
3	NUSA TENGGARA TIMUR	15
4	SULAWESI BARAT	15
5	SULAWESI TENGGARA	15
6	SULAWESI UTARA	15
7	SUMATERA SELATAN	14
8	ACEH	14
9	KALIMANTAN UTARA	14
10	PAPUA	14

Similarly, shipping delay can be defined in 2 ways:

1. The difference between pickup date and pickup limit date

```
SELECT
    customer_state,
    ROUND(AVG(shipping_delay / 60),2) AS [Average shipping de
lay (hours)]
FROM(
    SELECT
        DISTINCT f.order_key,
        shipping_delay,
        customer_state
    FROM
        Orders_Fact f
    JOIN
        [Order] o
    ON
        f.order_key = o.order_key
    JOIN
        Customer c
    ON
        f.customer_key = c.customer_key
) S
GROUP BY
    customer_state
ORDER BY
    [Average shipping delay (hours)] DESC
```

Result:

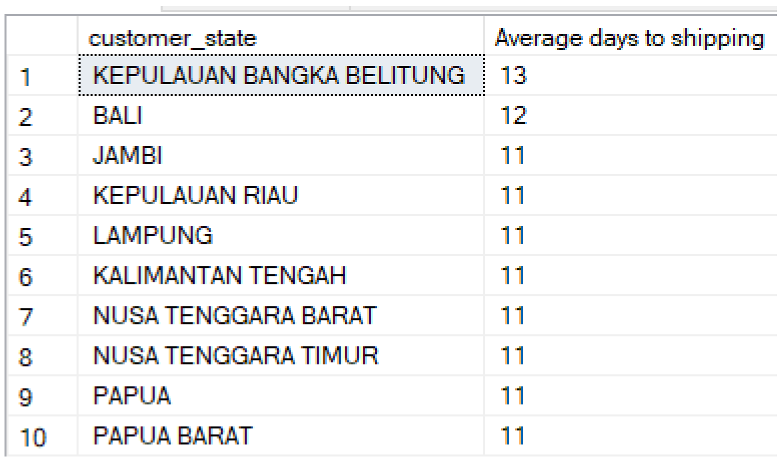


	customer_state	Average shipping delay (hours)
1	SULAWESI BARAT	12.42
2	KEPULAUAN BANGKA BELITUNG	10.51
3	MALUKU	9.75
4	KALIMANTAN TENGAH	7.95
5	BALI	7.53
6	SULAWESI UTARA	7.46
7	DKI JAKARTA	7.43
8	RIAU	7.15
9	SULAWESI TENGGARA	7.11
10	SUMATERA UTARA	6.88
11	JAWA TENGAH	6.68
12	JAWA BARAT	6.67
13	JAWA TIMUR	6.65
14	KALIMANTAN TIMUR	6.62
15	LAMPUNG	6.59

2. The difference between pickup date and delivery date

```
WITH order_dates AS (
    SELECT
        DISTINCT order_key,
        customer_state,
        CAST(d1.date AS DATE) AS pickup_date,
        CAST(d2.date AS DATE) AS delivery_date
    FROM
        Orders_Fact f
    JOIN Date d1
        ON d1.date_key = f.pickup_date_key
    JOIN Date d2
        ON d2.date_key = f.delivered_date_key
    JOIN Customer c
        ON c.customer_key = f.customer_key
    WHERE d1.date != 'unknown' AND d2.date != 'unknown'
)
SELECT
    customer_state,
    AVG(DATEDIFF(DAY,pickup_date, delivery_date)) AS [Average
days to shipping]
FROM
    order_dates
GROUP BY
    customer_state
ORDER BY
    [Average days to shipping] DESC
```

Result:



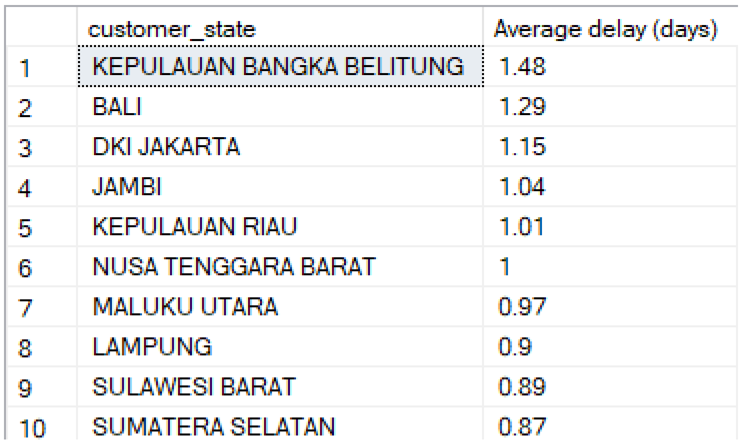
	customer_state	Average days to shipping
1	KEPULAUAN BANGKA BELITUNG	13
2	BALI	12
3	JAMBI	11
4	KEPULAUAN RIAU	11
5	LAMPUNG	11
6	KALIMANTAN TENGAH	11
7	NUSA TENGGARA BARAT	11
8	NUSA TENGGARA TIMUR	11
9	PAPUA	11
10	PAPUA BARAT	11

How long are the difference between estimated delivery time and actual delivery time in each state?

```
SELECT
  customer_state,
  ROUND(AVG(CAST(delivery_delay AS FLOAT)),2) AS [Average del
ay (days)]
FROM(
  SELECT
      DISTINCT f.order_key,delivery_delay ,customer_state
  FROM
      Orders_Fact f
  JOIN
      [Order] o
  ON
```

```
        f.order_key = o.order_key
    JOIN
        Customer c
    ON
        f.customer_key = c.customer_key
) S
GROUP BY
    customer_state
ORDER BY
    [Average delay (days)] DESC
```

Result:



	customer_state	Average delay (days)
1	KEPULAUAN BANGKA BELITUNG	1.48
2	BALI	1.29
3	DKI JAKARTA	1.15
4	JAMBI	1.04
5	KEPULAUAN RIAU	1.01
6	NUSA TENGGARA BARAT	1
7	MALUKU UTARA	0.97
8	LAMPUNG	0.9
9	SULAWESI BARAT	0.89
10	SUMATERA SELATAN	0.87