**DQL compared to SQL ,Splunk**

SQL (Structured Query Language) is used to query relational databases like MySQL, PostgreSQL.

Splunk SPL (Search Processing Language) is used for searching, filtering, and analyzing machine data in Splunk.

DQL (Dynatrace Query Language) is used in Dynatrace to query observability data (logs, metrics, traces).

**Loading Data for Querying**

**📌 Goal: Retrieve all records from the events dataset.**

sourcetype = event\*

SELECT \* FROM events

fetch events

**2️.Filtering Data (WHERE Clause Equivalent)**

📌 **Goal:** Retrieve events where event.type = travel.funnel.booking-payment.

sourcetype = event\* | where "event.type" = "travel.funnel.booking-payment"

SELECT \* FROM events WHERE 'event.type'="travel.funnel.booking-payment"

fetch events

| filter event.type == "travel.funnel.booking-payment"

**Multiple Filtering**

Goal: Retrieve events where:

* event.type = travel.funnel.booking-payment
* loyaltyStatus = Platinum
* childrenTravelers > 0

sourcetype = event\*

| where event.type = "travel.funnel.booking-payment" AND loyaltyStatus = "Platinum" AND childrenTravelers > 0

SELECT \* FROM events WHERE 'event.type'="travel.funnel.booking-payment" AND loyaltyStatus = "Platinum" AND childrenTravelers > 0

fetch events

| filter event.type == "travel.funnel.booking-payment" and loyaltyStatus == "Platinum" and childrenTravelers > 0

**Selecting Specific Fields**

**📌 Goal: Retrieve only the product field for travel.funnel.booking-payment events.**

sourcetype = event\*

| where event.type = "travel.funnel.booking-payment"

| fields product

SELECT product FROM events WHERE 'event.type'="travel.funnel.booking-payment"

fetch events

| filter event.type == "travel.funnel.booking-payment"

| fields product

A screenshot of a travel agency

AI-generated content may be incorrect.

Sorting is used to arrange data in ascending or descending order.

📌 **Sort events based on amount in descending order**

fetch events

| sort amount desc

**Calculations and sorting**

sourcetype = event\*

| where event.type = "travel.funnel.booking-payment"

| eval journeyweeks = journeyDuration/7

| sort -journeyweeks

SELECT journeyDuration/7 AS journeyWeeks FROM events WHERE 'event.type'="travel.funnel.booking-payment" ORDER BY journeyWeeks DESC

fetch event

| filter event.type == "travel.funnel.booking-payment"

| fieldsAdd journeyWeeks = journeyDuration/7

| sort journeyWeeks desc

**Grouping**

**Grouping helps us organize data based on a field (e.g., event.type, travelAgency).**

A screenshot of a travel agency

AI-generated content may be incorrect.

**📌 Find unique travel agencies (without duplicates)**

fetch events

| summarize count(), by:travelAgency

* **summarize count(), by:travelAgency → Groups events by travel agency and counts the number of events per agency.**

**A screenshot of a phone

AI-generated content may be incorrect.**

sourcetype = event\*

| stats by "event.type"

SELECT DISTINCT 'event.type' FROM events

fetch events

| summarize count(), by:event.type

| fields event.type

==Other examples==

| summarize count()

| summarize event\_count = count()

| summarize event\_count = count(), by:{country=client.loc\_cc, customer}

**Aggregation**

After grouping selected records based on a field, we can aggregate the results to a new output.

A screenshot of a travel agency

AI-generated content may be incorrect.

Aggregation helps us **calculate summaries** like SUM, AVG, COUNT, MAX, MIN.

📌 **Find total revenue (sum(amount)) per travelAgency**

fetch events

| summarize total\_revenue = sum(amount), by:travelAgency

* sum(amount) → Adds up all amount values **per travel agency**.
* by:travelAgency → Groups results by travelAgency.

A white rectangular object with a black background

AI-generated content may be incorrect.

📌 **Find the highest amount (max transaction) per travelAgency**

fetch events

| summarize max\_amount = max(amount), by:travelAgency

* max(amount) → Finds **highest** amount per travel agency.
* by:travelAgency → Groups by travelAgency.

📌 Find the average journey duration (avg(journeyDuration)) per event.type

fetch events

| summarize avg\_duration = avg(journeyDuration), by:event.type

* avg(journeyDuration) → Finds **average duration** per event type.
* by:event.type → Groups by event.type.

**Combining Sorting + Grouping + Aggregation**

A screenshot of a travel agency

AI-generated content may be incorrect.

📌 **Find total revenue (sum(amount)) per travelAgency, sorted in descending order**

fetch events

| summarize total\_revenue = sum(amount), by:travelAgency

| sort total\_revenue desc

* summarize sum(amount), by:travelAgency → Groups revenue by travel agency.
* sort total\_revenue desc → Sorts **from highest to lowest revenue**.

fetch events

| filter event.type == "travel.funnel.booking-payment"

| summarize sum = sum(amount), by:travelAgency