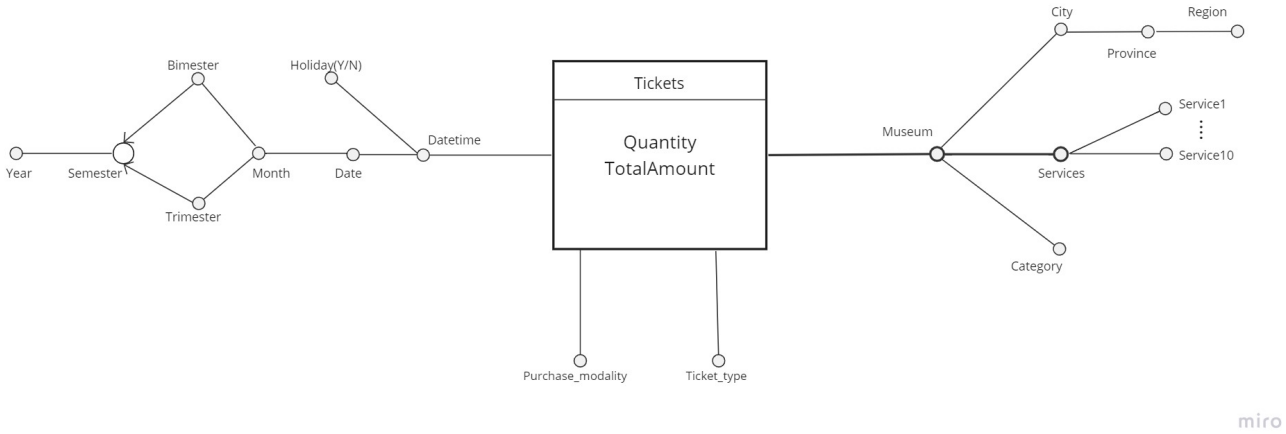
Data Science and Data Base Technologies - Homework 1 - StudentID s332135

Exercise 1

Conceptual schema



Logical schema

Fact Table

Tickets(MuseumID, TimeID, PurchaseInfoID, Quantity, TotalAmount)

Dimension table

Museum(MuseumID, Museum, Category, CityID, ServicesID)
City(CityID, City, Province, Region)
Services(ServicesID, Service1, Service2, Service3, Service4, Service5, Service6, Service7, Service8, Service9, Service10)
PurchaseInfo(PurchaseInfoID, Purchase_modality, Ticket_type)
Time(TimeID, DateTime, Date, Holiday, Month, Bimester, Trimester, Semester, Year)

Exercise 2

2.1

Separately for each ticket type and for each month (of the ticket validity), analyze:
the average daily revenue, the cumulative revenue from the beginning of the year,

the percentage of tickets related to the considered ticket type over the total number of tickets of the month

```
SELECT Ticket_type, Month,
    SUM(TotalAmount)/ COUNT(DISTINCT Date),
    SUM(SUM(TotalAmount)) OVER (PARTITION BY Year
                ORDER BY Month ROWS UNBOUNDED PRECEDING) AS CumulativeForYear,
    SUM(Quantity)/SUM(SUM(Quantity)) OVER (PARTITION BY Month) * 100 AS Percentage
FROM Tickets t, Time tm, PurchaseInfo p
WHERE t.TimeID = tm.TimeID and t.PurchaseInfoID = p.PurchaseInfoID
GROUP BY Ticket_type, Month, Year
ORDER BY Month, Ticket_type, Year;
```

2.2

Considering the ticket of 2021. Separately for each museum and ticket type analyze: the average revenue for a ticket, the percentage of revenue over the total revenue for the corresponding museum category, assign a rank to the museum, for each ticket type, according to the total number of tickets in decreasing order.

```
SELECT t.MuseumID, Ticket_type,
    SUM(TotalAmount)/SUM(Quantity) AS AverageRevenueForTicket,
    SUM(TotalAmount)/SUM(SUM(TotalAmount)) OVER (PARTITION BY Category) * 100,
    RANK() OVER (PARTITION BY Ticket_type ORDER BY SUM(Quantity) DESC) AS ranking
FROM Tickets t, Museum m, PurchaseInfo p, Time tm
WHERE t.MuseumID = m.MuseumID AND t.PurchaseInfoID = p.PurchaseInfoID AND tm.TimeID
= t.TimeID AND Year = 2021
GROUP BY t.MuseumID, Ticket_type, Category
ORDER BY Ranking, Ticket_type, MuseumID
```

Exercise 3

1

Analyze the average monthly revenue related to each ticket type and for each semester.

```
SELECT Ticket_type, Semester, SUM(TotalAmount) / COUNT(DISTINCT Month)
FROM Tickets t, Time tm, PurchaseInfo p
WHERE t.PurchaseInfoID = p.PurchaseInfoID AND t.TimeID = tm.TimeID
GROUP BY Ticket_type, Semester, Month
ORDER BY Ticket_Type, Month
```

2

Separately for each ticket type and for each month analyze the cumulative revenue from the beginning of the year.

```
SELECT Ticket_type, Year,
               SUM(SUM(TotalAmount)) OVER (PARTITION BY Year ORDER BY Month
                       ROWS UNBOUNDED PRECEDING)
FROM Tickets t, Time tm, PurchaseInfo p
WHERE t.PurchaseInfoID = p.PurchaseInfoID AND t.TimeID = tm.TimeID
GROUP BY Ticket_type, Month
ORDER BY Ticket_Type, Month
```

3

Considering only the tickets purchased online, separately for each ticket type and for each month analyze the total number of tickets, the total revenue and the average revenue

```
SELECT Ticket_type, Month,
               SUM(Quantity),
               SUM(TotalAmount),
               SUM(TotalAmount)/SUM(Quantity)
FROM Tickets t, Time tm, PurchaseInfo p
WHERE t.PurchaseInfoID = p.PurchaseInfoID AND t.TimeID = tm.TimeID AND
p.PurchaseModality = "Online"
GROUP BY Ticket_type, Month
ORDER BY Ticket_type, Month
```

4

Separately for each ticket type and for each month analyze the total number of tickets, the total revenue and the average revenue for year 2021.

```
SELECT Ticket_type, Month,
               SUM(Quantity),
               SUM(TotalAmount),
               SUM(TotalAmount)/SUM(Quantity)
FROM Tickets t, Time tm, PurchaseInfo p
WHERE t.PurchaseInfoID = p.PurchaseInfoID AND t.TimeID = tm.TimeID AND tm.Year=2021
GROUP BY Ticket_type, Month
ORDER BY Ticket_type, Month
```

5

Analyze the percentage of tickets related to each ticket type and month over the total number of tickets of the month.

```
SELECT SUM(Quantity) / SUM(SUM(Quantity)) OVER (PARTITION BY Month) * 100
FROM Tickets t, Time tm, PurchaseInfo p
WHERE t.PurchaseInfoID = p.PurchaseInfoID AND t.TimeID = tm.TimeID
GROUP BY Ticket_type, Month
ORDER BY Ticket_Type, Month
```

3.1

```
CREATE MATERIALIZED VIEW MAT_VIEW AS
        SELECT Ticket_type, Semester, Month, Year, PurchaseModality,
                SUM(Quantity),
                SUM(TotalAmount),
        FROM Tickets t, Time tm, PurchaseInfo p,
        WHERE t.TimeID = tm.TimeID AND t.PurchaseInfoID = p.PurchaseInfoID
        GROUP BY Ticket_type, Semester, Month, Year, PurchaseModality
```

3.2

We consider only the tables and attributes involved in the materialized view.

```
CREATE MATERIALIZED VIEW LOG ON Tickets
WITH SEQUENCE, ROWID
(TimeID, PurchaseInfoID, Quantity, TotalAmount)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON Time
WITH SEQUENCE, ROWID
(TimeID, Month, Semester, Year)
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON PurchaseInfo
WITH SEQUENCE, ROWID
(PurchaseInfoID, Purchase_modality, Ticket_type)
INCLUDING NEW VALUES;
```

3.3

Operations as Insert and Update on Time, PurchaseInfo and Tickets tables cause update of the materialized view.

Exercise 4

4.1

```
CREATE TABLE VM1(
        Ticket_type INTEGER NOT NULL
        ,Semester VARCHAR(6) NOT NULL
        ,Month VARCHAR(7) NOT NULL
        ,Year INTEGER NOT NULL
        ,PurchaseModality VARCHAR(30) NOT NULL
        ,Quantity INTEGER NOT NULL
        ,TotalAmount INTEGER NOT NULL
        ,PRIMARY KEY(Ticket_type, Purchase_Modality, Month)
)
```

4.2

```
INSERT INTO VM1(Ticket_type, Semester, Month, Year, PurchaseModality,Quantity,
TotalAmount)
        (SELECT Ticket_type, Semester, Month, Year, PurchaseModality,
                SUM(Quantity),
                SUM(TotalAmount),
        FROM Tickets t, Time tm, PurchaseInfo p,
        WHERE t.TimeID = tm.TimeID AND t.PurchaseInfoID = p.PurchaseInfoID
        GROUP BY Ticket_type, Semester, Month, Year, PurchaseModality);
```

4.3

```
CREATE TRIGGER RefreshViewRevenue
AFTER INSERT ON Tickets
FOR EACH ROW
DECLARE
N number;
varTicket_type INTEGER, varSemester VARCHAR(6), varMonth VARCHAR(7), varYear
INTEGER, varPurchaseModality VARCHAR(30);

BEGIN
SELECT Semester, Month, Year INTO varSemester, varMonth, varYear
FROM Time
```

```
WHERE TimeID = :NEW.TimeID:;

SELECT Ticket_type, PurchaseModality INTO varTicket_type, varPurchaseModality
FROM PurchaseInfo
WHERE PurchaseInfoID = :NEW.PurchaseInfoID:;

SELECT COUNT(*) INTO N
FROM VM1
WHERE Month = varMonth AND Ticket_type = vatTicket_type AND PurchaseModality =
varPurchaseModality
IF (N > 0) THEN
        UPDATE VM1
        SET TotalAmount = TotalAmount + :NEW.TotalAmount
                Quantity = Quantity + :NEW.Quantity
        WHERE Month = varMonth AND Ticket_type = vatTicket_type AND
PurchaseModality = varPurchaseModality;
ELSE
        INSERT INTO VM1(Ticket_type, Semester, Month, Year, PurchaseModality,
TotalAmount, Quantity)
        VALUES (varTicket_type, varSemester, varMonth, varYear,
varPurchaseModality, :NEW.TotalAmount, :NEW.Quantity);
END IF;
END;
```

4.4

The trigger triggers after an insertion query is executed on the fact table.