

Group 1 report
CSIP5102 (Data Warehouse Design)
2023-2024

Group Members

Name	Student ID	Mail
Soubhi Saad	P2739759	p2739759@my365.dmu.ac.uk
Yaser Basheer	P2810257	p2810257@my365.dmu.ac.uk
Jerin Tom	P2797471	p2797471@my365.dmu.ac.uk
Mathieu Charance	P2739646	p2739646@my365.dmu.ac.uk
Abdullah Ahsin	P2600762	p2600762@my365.dmu.ac.uk
Alan Rodrigues	P2809461	p2809461@my365.dmu.ac.uk
Iftekhirul	P2777638	p2777638@my365.dmu.ac.uk
David Banda	P17155768	p17155768@my365.dmu.ac.uk

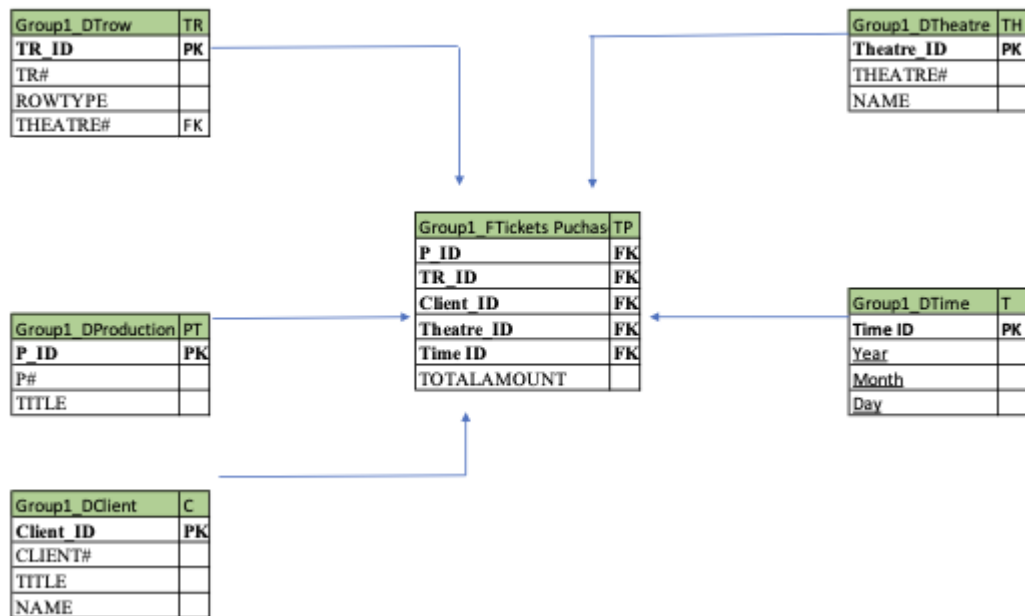
Summary

1. Star schema design.....	2
1.1 Star schema.....	2
1.2 Explanation on your star schema.....	2
2. Derivation of logical relations.....	3
2.1 A list of logical relations (it has to follow the formal representation as our lecture notes) (ALAN).....	3
2.2 Explanation on the mapping from your star schema to the logical relations.....	3
3. Creation of tables with integrity rules.....	5
3.1 SQL code for each table creation (Ifte & Abdullah).....	5
3.2 Evidence of the code working and table created (small screenshots) (Ifte & Abdullah).....	5
4. Data source identification, data extraction, transformation, and loading (Soubhi & Mathieu).....	7
4.1 Data source mapping.....	7
4.2 ETL code.....	8
4.3 Evidence of ETL code works.....	12
5. Data analysis (Jerry & Yaser).....	14
5.1 SQL code for the required queries (both Data Mart and the relational model).....	14
Production-wise reporting of daily analysis :.....	14
The title and total sale of each production :.....	14
The clients visited MT in at least 6 different months in a year.....	16
The number of clients who have only visited MT theatres in one single month and no visit in any other month :.....	16
The month of the year with the highest ticket sale for a theatre each year :.....	16
The most popular row type in each theatre by value of ticket sale :.....	17
5.2 Evidence of the SQL code working and their results.....	17
Production-wise reporting of daily analysis :.....	17
The title and total sale of each production :.....	18
The theatre name and total yearly sale for each theatre :.....	18
The most popular row type in each theatre by value of ticket sale :.....	19
6. Justification of the data mart design and comparison of data mart and OLTP (David) 20	
6.1 Please compare your solutions in 5 and comment on how they satisfy the requirements of the data analysis. Furthermore, please compare Data Mart and the relational model in data analysis in general. (David).....	20
7. Critical analysis and reflection.....	21
7.1 Please give a critical analysis on your group's work, what is your success, what is not as expected. How is your group collaboration. With more time, what could be further improved. The management evaluation report from each member will have impact on this part. (Mathieu).....	21
Management Evaluation Report.....	22
Appendix.....	23

1. Star schema design

1.1 Star schema

Please draw a star schema here: (Alan)



1.2 Explanation on your star schema

a. What are the fact and dimensions and why they are chosen

Dimensions

1	TR for group1_DTrow This dimension is required to analyze most popular row type in each theatre by value of ticket sale
2	PT for group1_DProduction This dimension is required to analyze the production wise reporting of daily analysis
3	C for group1_DClient This dimension is required to analyze the number of clients visited MT in at least 6 different months in a year
4	TH for group1_DTheatre This dimension is required to analyze the theatre name and total yearly sale for each theatre.
5	T for group1_DTime This Dimension is required to analyze daily, monthly, and yearly sales

Fact Table - TP for group1_FTicketPurchase

This Ticket sales table is chosen as the fact table as it satisfies the minimum criteria required to perform the analysis based on the questions. All the data present in the fact table are quantifiable, which is the requirement of a fact table.

b. Why their attributes are chosen

As we have illustrated in the star schema above, we have chosen only those attributes that will be used to make the required analysts, because our aim is to create a data warehouse and optimize everything so as to have an environment where data can be analyzed efficiently. Well-chosen attributes can improve query performance, not to mention the fact that choosing well-normalized attributes helps to avoid database anomalies, making our work error-free. If we use all the attributes, we use storage space for which there is ultimately no point in using them, which could have an impact on performance if we have millions of rows of data, so we need to take only what we need as attributes.

c. What is the granularity

The granularity or grain of a fact table is very important, as it allows us to understand the nature of the different facts. So, in our case, the granularity, for each row in group1_FTicketPurchase it represents the ticket sales by Midlands Theater chain for client for a month, a year, or a day.

Granularity is an important element of data warehousing, because it affects the level of detail that we can provide in our reports and the performance of the queries.

2. Derivation of logical relations

2.1 A list of logical relations (it has to follow the formal representation as our lecture notes)

TR - group1_DTrow	(TR_ID , TR#, ROWTYPE, THEATRE#) THEATRE# is the foreign key to Theatre
PT - group1_DProduction	(P_ID , P#, TITLE)
C - group1_DClient	(CLIENT_ID , CLIENT#, TITLE, NAME)
TH - group1_DTheatre	(THEATRE_ID , THEATRE#, NAME)
T - group1_DTime	(TimeID , Year, Month, Day)
TP(Fact Table) - group1_FTicketsPurchase	(P_ID , TR_ID , CLIENT_ID , THEATRE_ID , Time ID , TOTAL AMOUNT)

2.2 Explanation on the mapping from your star schema to the logical relations

a. How is the logical relations mapped from your star schema (which relations is corresponding to which entity in your star schema)

TR_ID is the primary key in Trow, **P_ID** is the primary key in the production table, **Theatre_ID** is the primary key in Theatre, **Time ID** is the primary key in Time.

In the Fact table, P_ID is the foreign key referring to Production, TR_ID is the foreign key referring to Trow, Client_ID is the foreign key referring to Client, Theatre_ID is the foreign key referring to Theatre and Time ID is the foreign key referring to Time.

P_ID, TR_ID, CLIENT_ID, THEATRE_ID, Time ID is the composite key.

P_ID, TR_ID, CLIENT_ID, THEATRE_ID, TIME_ID surrogate keys.

P#, TR#, CLIENT#, THEATRE# are natural keys

- b. What are the Primary Keys (PK) and Foreign Keys (FK) in the fact and dimensions. For each FK, what does it refer to ? (ALAN)

TR - group1_DTrow	TR# is the primary key
PT - group1_DProduction	P# is the primary key
C - group1_DClient	CLIENT# is the primary key
TH - group1_DTheatre	THEATRE# is the primary key
T - group1_DTime	TimeID is the primary key
TP(Fact Table) - group1_FTickets Purchase	P# is the foreign key TR# is the foreign key CLIENT# is the foreign key THEATRE# is the foreign key Time ID is the foreign key P#, TR#, CLIENT#, THEATRE#, Time ID is the composite key

3. Creation of tables with integrity rules

For each table:

3.1 SQL code for each table creation

3.2 Evidence of the code working and table created (small screenshots)

Create Table **group1_Dtrow**

```
1  create table group1_Dtrow (
2  TR_ID NUMBER(5) PRIMARY KEY,
3  TR# NUMBER(5) NOT NULL,
4  ROWTYPE VARCHAR2(10),
5  THEATRE# NUMBER(5))
```

Results	Explain	Describe	Saved SQL	His
Table created.				

Create Table **group1_Dclient**

```
1  CREATE TABLE group1_DCLIENT (
2  CLIENT_ID NUMBER (5) PRIMARY KEY,
3  CLIENT# NUMBER(5) NOT NULL,
4  TITLE VARCHAR2(20) NOT NULL,
5  NAME VARCHAR2(20) NOT NULL)
6
```

Results	Explain	Describe	Saved SQL	His
Table created.				

Create Table **group1_Dtheatre**

```
1 CREATE TABLE group1_DTHEATRE (
2 THEATRE_ID NUMBER (5) PRIMARY KEY,
3 THEATRE# NUMBER (5) NOT NULL,
4 NAME VARCHAR2(20) NOT NULL);
```

Results	Explain	Describe	Saved SQL	His
---------	---------	----------	-----------	-----

Table created.

Create Table **group1_Dproduction**

```
1 CREATE TABLE group1_DPRODUCTION (
2 P_ID NUMBER(5) PRIMARY KEY,
3 P# NUMBER (5) NOT NULL,
4 TITLE VARCHAR2(20) NOT NULL);
```

Results	Explain	Describe	Saved SQL	His
---------	---------	----------	-----------	-----

Table created.

Create Table **group1_Dtime**

```
1 CREATE TABLE group1_DTIME (
2 TIME_ID NUMBER (5) PRIMARY KEY,
3 YEAR number(4) NOT NULL,
4 MONTH NUMBER(2) NOT NULL,
5 DAY NUMBER(2) NOT NULL);
```

Results	Explain	Describe	Saved SQL	His
---------	---------	----------	-----------	-----

Table created.

Create Table **group1_Fticketpurchase (FACT TABLE)**

```

1 CREATE TABLE group1_FTICKETPURCHASE (
2 THEATRE_ID NUMBER(5) CONSTRAINT fk22 REFERENCES group1_DTHEATRE,
3 P_ID NUMBER(5) CONSTRAINT fk23 REFERENCES group1_DPRODUCTION,
4 TR_ID NUMBER(5) CONSTRAINT fk24 REFERENCES group1_DTROW,
5 CLIENT_ID NUMBER(5) CONSTRAINT fk25 REFERENCES group1_DCLIENT,
6 TIME_ID NUMBER(5) CONSTRAINT fk26 REFERENCES group1_DTIME,
7 TOTALAMOUNT NUMBER(10,2) NOT NULL,
8 CONSTRAINT pk27 PRIMARY KEY (THEATRE_ID, P_ID, TR_ID, CLIENT_ID, TIME_ID));

```

Results Explain Describe Saved SQL History

Table created.

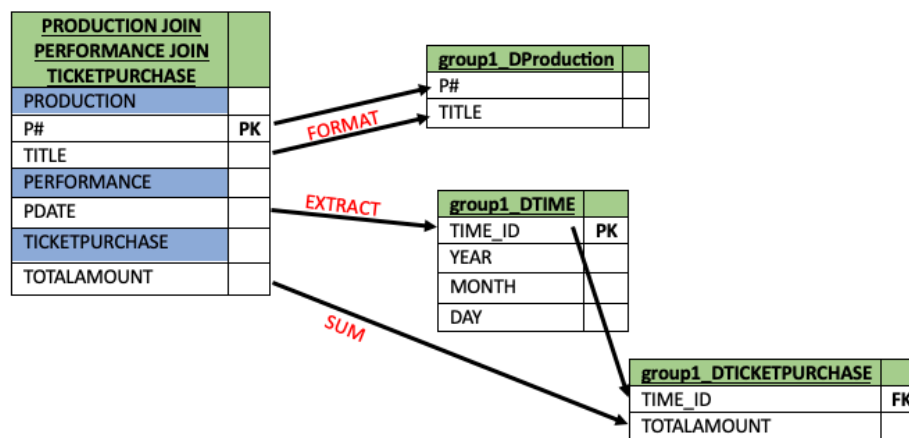
4. Data source identification, data extraction, transformation, and loading

4.1 Data source mapping

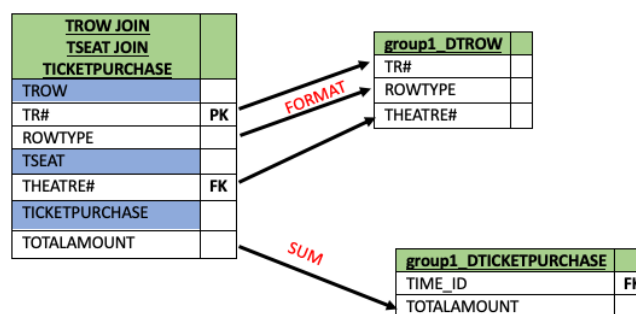
Please draw the map between data source and their destination in your Data Mart

For each ETL operation:

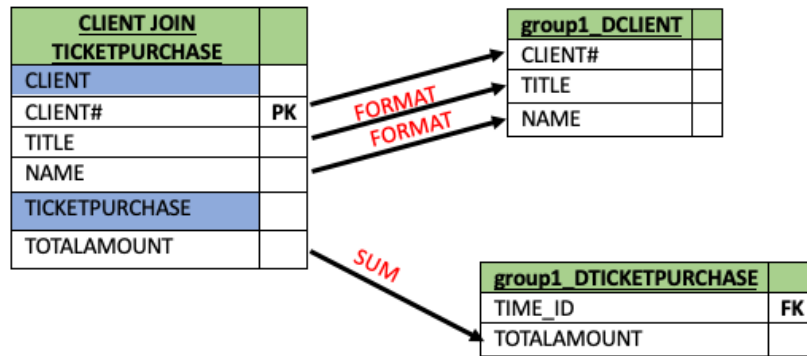
group1_DPRODUCTION



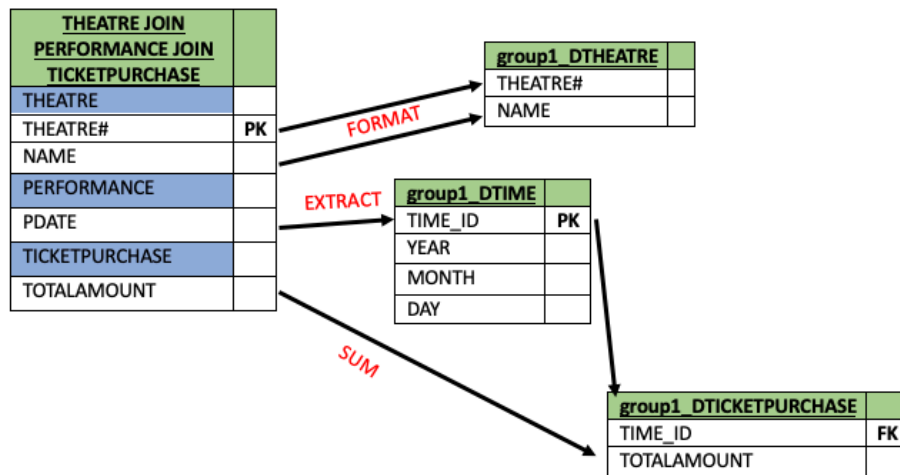
group1_DTROW



group1_DCLIENT



group1_DTHEATRE



4.2 ETL code

Data source mapping represents the ETL process between the OLTP data and the data mart. Before performing ETL, it is necessary to create sequences. These sequences will produce surrogate keys to guarantee the unicity of the different dimension tables in our data mart. The screenshots below show the sequence creation with the ETL process (**ALL THE CODE WILL BE AT THE END OF THE APPENDIX**) :

group1_DPRODUCTION SURROGATE KEYS

```

1 CREATE SEQUENCE DP_seq START WITH 1
2 INCREMENT BY 1
3 NOCACHE
4 NOCYCLE;

```

Results Explain Describe Saved SQL History

Sequence created.

group1_DTROW

SURROGET KEYS

```
1 CREATE SEQUENCE DTR_seq START WITH 1
2 INCREMENT BY 1
3 NOCACHE
4 NOCYCLE;
5
```

Results Explain Describe Saved SQL History

Sequence created.

group1_DCLIENT SURROGET KEYS

```
1 CREATE SEQUENCE DCLIENT_seq START WITH 1
2 INCREMENT BY 1
3 NOCACHE
4 NOCYCLE;
```

Results Explain Describe Saved SQL History

Sequence created.

group1_DTHEATRE SURROGET KEYS

```
1 CREATE SEQUENCE DTHEATRE_seq START WITH 1
2 INCREMENT BY 1
3 NOCACHE
4 NOCYCLE;
```

Results Explain Describe Saved SQL History Bottom S

Sequence created.

group1_DTIME SURROGET KEYS

```
1 CREATE SEQUENCE DTIME_seq START WITH 1
2 INCREMENT BY 1
3 NOCACHE
4 NOCYCLE;
```

group1_DPRODUCTION ETL

```
1  INSERT INTO group1_DPRODUCTION
2  SELECT DP_seq.nextval, P#, TITLE
3  FROM (SELECT DISTINCT P#, UPPER (TRIM(TITLE)) TITLE
4  FROM (SELECT P.P#, P.TITLE
5  FROM ops$yyang00.production p,
6  ops$yyang00.PERFORMANCE PER,
7  ops$yyang00.TICKETPURCHASE TP
8  WHERE P.P# = PER.P# AND TP.PER# = PER.PER#)) ;
```

Results

Explain

Describe

Saved SQL

History

78 row(s) inserted.

group1_DTROW ETL

```
1  INSERT INTO group1_DTROW
2  SELECT DTR_seq.nextval, TR#, ROWTYPE, THEATRE#
3  FROM (SELECT DISTINCT UPPER (TRIM(ROWTYPE)) ROWTYPE, TR#, THEATRE#
4  FROM (SELECT Otr.TR#, Otr.ROWTYPE, OTR.THEATRE# FROM ops$yyang00.TROW OTR, ops$yyang00.TSEAT TS
5  WHERE OTR.TR# = TS.TR#));
```

Results

Explain

Describe

Saved SQL

History

32 row(s) inserted.

group1_DCLIENT ETL

```

1  INSERT INTO group1_DCLIENT
2  SELECT DCLIENT_seq.nextval, CLIENT#, TITLE, NAME
3  FROM (SELECT DISTINCT CLIENT#, UPPER (TRIM(TITLE)) TITLE, UPPER(TRIM(NAME)) NAME
4  FROM (SELECT C.CLIENT#, C.TITLE, C.NAME FROM ops$yyang00.CLIENT C, ops$yyang00.TICKETPURCHASE TP WHERE C.CLIENT# = TP.CLIENT#));

```

Results Explain Describe Saved SQL History

4978 row(s) inserted.

group1_DTIME ETL

```

1  INSERT INTO group1_DTIME
2  SELECT DTIME_seq.nextval, YEAR, MONTH, DAY
3  FROM (SELECT DISTINCT
4  EXTRACT (Year FROM PDATE) Year,
5  EXTRACT (Month FROM PDATE) Month,
6  EXTRACT (DAY FROM PDATE) DAY
7  FROM ops$yyang00.PERFORMANCE);

```

Results Explain Describe Saved SQL History

4978 row(s) inserted.

group1_FTICKETPURCHASE (FACT TABLE)

```

123
124  INSERT INTO group1_FTICKETPURCHASE
125  SELECT THEATRE_ID, P_ID, TR_ID, CLIENT_ID, TIME_ID, TOTALAMOUNT
126  FROM (
127  -- SELECT
128  -- T.THEATRE_ID, P.P_ID, C.CLIENT_ID, TM.TIME_ID, TR.TR_ID, SUM(TP.TOTALAMOUNT) AS TOTALAMOUNT
129  -- FROM
130  -- ops$yyang00.PERFORMANCE PER,
131  -- group1_DCLIENT C,
132  -- group1_DTHEATRE T,
133  -- group1_DPRODUCTION P,
134  -- group1_DTIME TM,
135  -- group1_DTRGW TR,
136  -- ops$yyang00.TICKETPURCHASE TP,
137  -- ops$yyang00.TSEAT TS
138  -- WHERE
139  -- PER.THEATRE# = T.THEATRE#
140  -- AND PER.P# = P.P#
141  -- AND TS.TR# = TR.TR#
142  -- AND TP.CLIENT# = C.CLIENT#
143  -- AND TP.PER# = PER.PER#
144  -- AND EXTRACT(YEAR FROM PER.PDATE) = TM.YEAR
145  -- AND EXTRACT(MONTH FROM PER.PDATE) = TM.MONTH
146  -- AND EXTRACT(DAY FROM PER.PDATE) = TM.DAY
147  -- GROUP BY
148  -- T.THEATRE_ID, P.P_ID, TR.TR_ID, C.CLIENT_ID, TM.TIME_ID
149  -- FETCH FIRST 27868 ROWS ONLY)

```

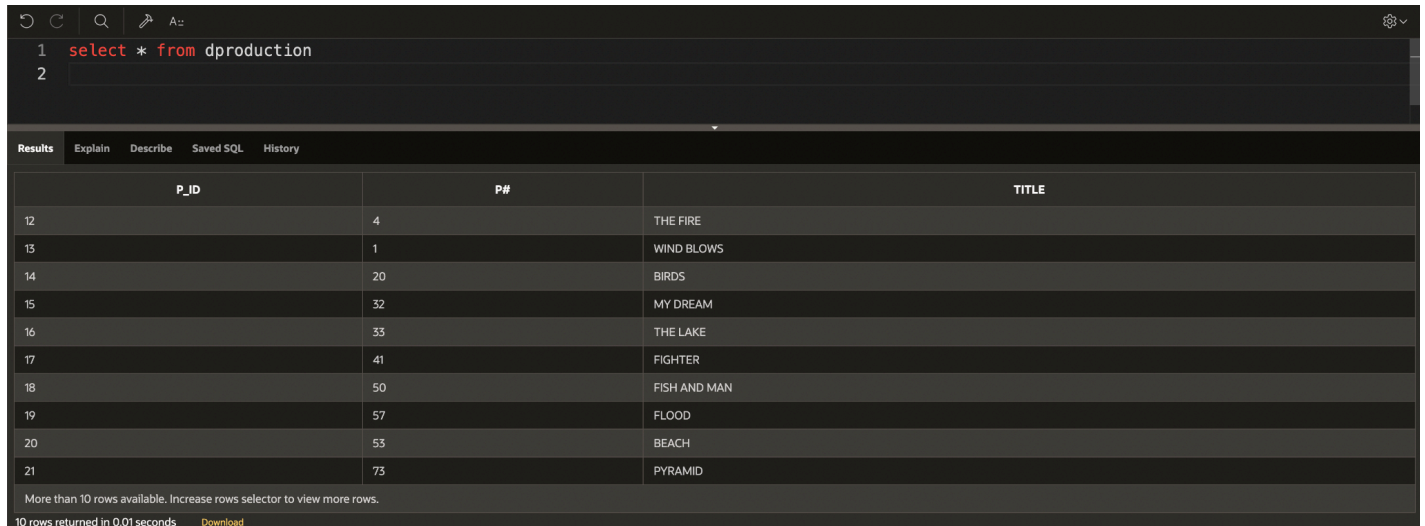
Results Explain Describe Saved SQL History

27868 row(s) inserted.

4.3 Evidence of ETL code works

Please provide the screenshot to show your code works and give the correct result. If the output is long, just show the last screen with the returned number of rows inserted.

group1_DPRODUCTION

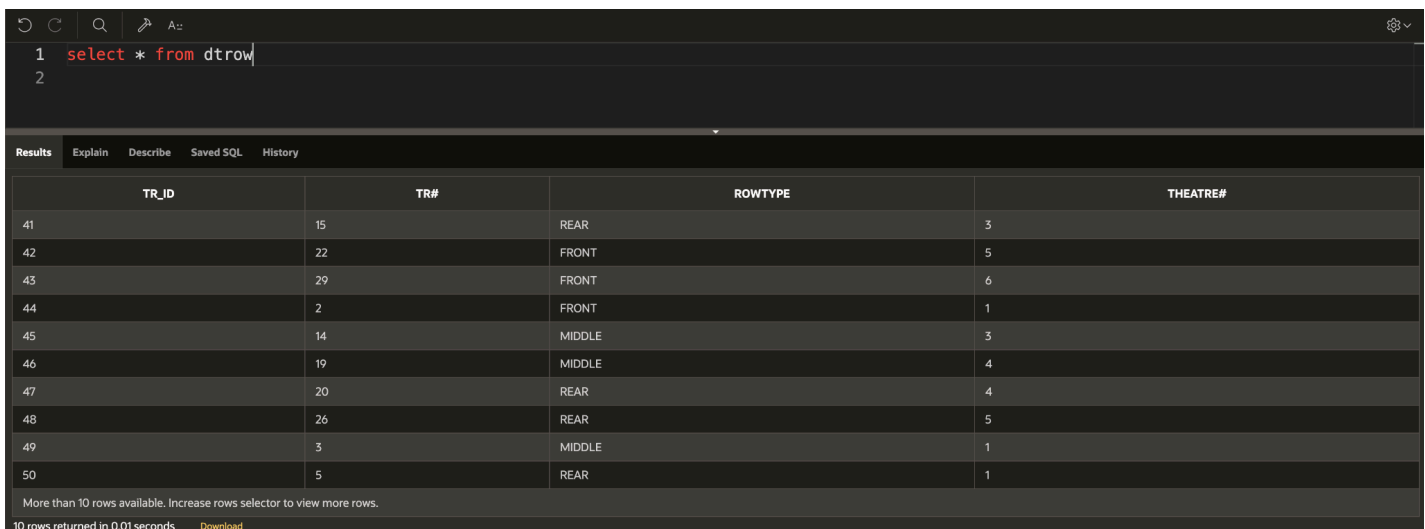


The screenshot shows a SQL query editor with the query `select * from dproduction` and its results. The results table has three columns: P_ID, P#, and TITLE. It displays 10 rows of data, with a message indicating that more than 10 rows are available.

P_ID	P#	TITLE
12	4	THE FIRE
13	1	WIND BLOWS
14	20	BIRDS
15	32	MY DREAM
16	33	THE LAKE
17	41	FIGHTER
18	50	FISH AND MAN
19	57	FLOOD
20	53	BEACH
21	73	PYRAMID

More than 10 rows available. Increase rows selector to view more rows.
10 rows returned in 0.01 seconds [Download](#)

group1_DTROW

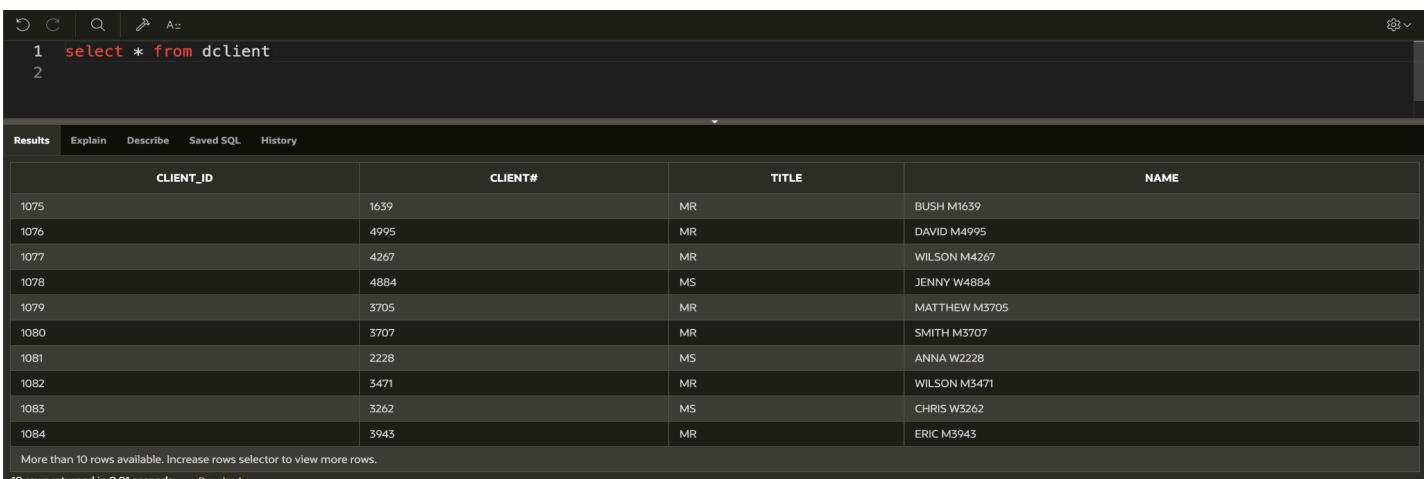


The screenshot shows a SQL query editor with the query `select * from dtrow` and its results. The results table has four columns: TR_ID, TR#, ROWTYPE, and THEATRE#. It displays 10 rows of data, with a message indicating that more than 10 rows are available.

TR_ID	TR#	ROWTYPE	THEATRE#
41	15	REAR	3
42	22	FRONT	5
43	29	FRONT	6
44	2	FRONT	1
45	14	MIDDLE	3
46	19	MIDDLE	4
47	20	REAR	4
48	26	REAR	5
49	3	MIDDLE	1
50	5	REAR	1

More than 10 rows available. Increase rows selector to view more rows.
10 rows returned in 0.01 seconds [Download](#)

group1_DCLIENT



The screenshot shows a SQL query editor with the query `select * from dclient` and its results. The results table has four columns: CLIENT_ID, CLIENT#, TITLE, and NAME. It displays 10 rows of data, with a message indicating that more than 10 rows are available.

CLIENT_ID	CLIENT#	TITLE	NAME
1075	1639	MR	BUSH M1639
1076	4995	MR	DAVID M4995
1077	4267	MR	WILSON M4267
1078	4884	MS	JENNY W4884
1079	3705	MR	MATTHEW M3705
1080	3707	MR	SMITH M3707
1081	2228	MS	ANNA W2228
1082	3471	MR	WILSON M3471
1083	3262	MS	CHRIS W3262
1084	3943	MR	ERIC M3943

More than 10 rows available. Increase rows selector to view more rows.
10 rows returned in 0.01 seconds [Download](#)

group1_DTHEATRE

1	select * from dtheatre
2	

Results	Explain	Describe	Saved SQL	History
THEATRE_ID	THEATRE#	NAME		
7	1	CROPSTON		
8	6	CLINTON		
9	5	BIRSTALL		
10	2	VICTORY		
11	3	BEESTON		
12	4	VUE		

6 rows returned in 0.00 seconds [Download](#)

group1_DTIME

</

group1_FTICKETPURCHASE (FACT)

5. Data analysis

5.1 SQL code for the required queries (both Data Mart and the relational model)

Production-wise reporting of daily analysis :

Data mart

```
SQL Commands

Language SQL Rows 10 Clear Command Find Tables

SELECT P.title as dproduction_title, dt.day, SUM(f.totalamount) as total_sales
FROM Group1_dproduction P, Group1_dtime dt, Group1_fticketpurchase f
WHERE P.p_id = f.p_id AND dt.time_id = f.time_id
GROUP BY P.title, dt.day;
```

Relational model

```
SELECT P.title as prod_title, EXTRACT(DAY from Per.PDATE), SUM(TP.totalamount) as Total_sale
FROM ops$yyang00.production P, ops$yyang00.performance PER, ops$yyang00.ticketpurchase TP
WHERE P.p# = Per.p# AND Per.per# = TP.Per#
GROUP BY P.title, EXTRACT(DAY from Per.pdate);
```

The title and total sale of each production :

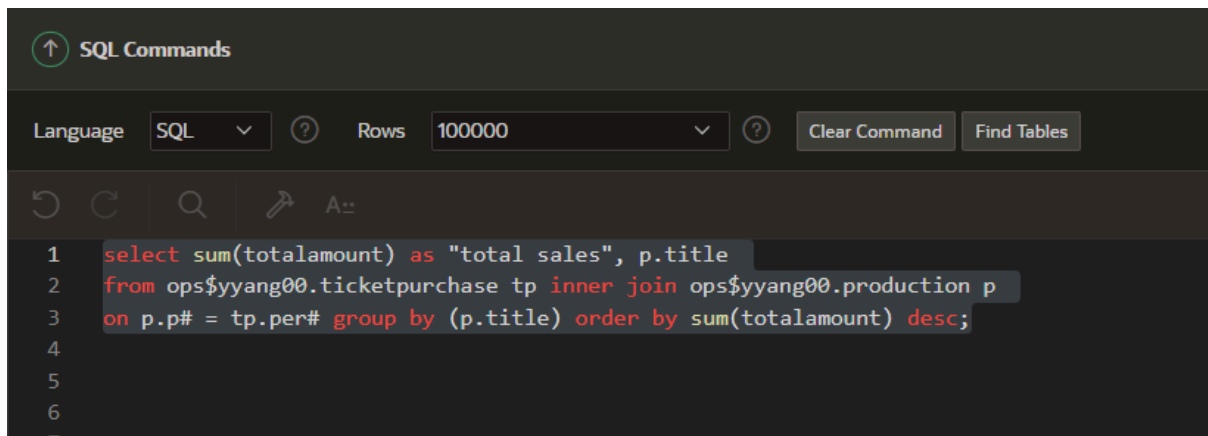
Data mart

```
SQL Commands

Language SQL Rows 10 Clear Command Find Tables

select sum(totalamount) as "total sales", p#.title
from Group1_fticketpurchase tp inner join Group1_dproduction p#
on p#.p_id=tp.p_id group by (p#.title) order by sum(totalamount) desc
```

Relational model

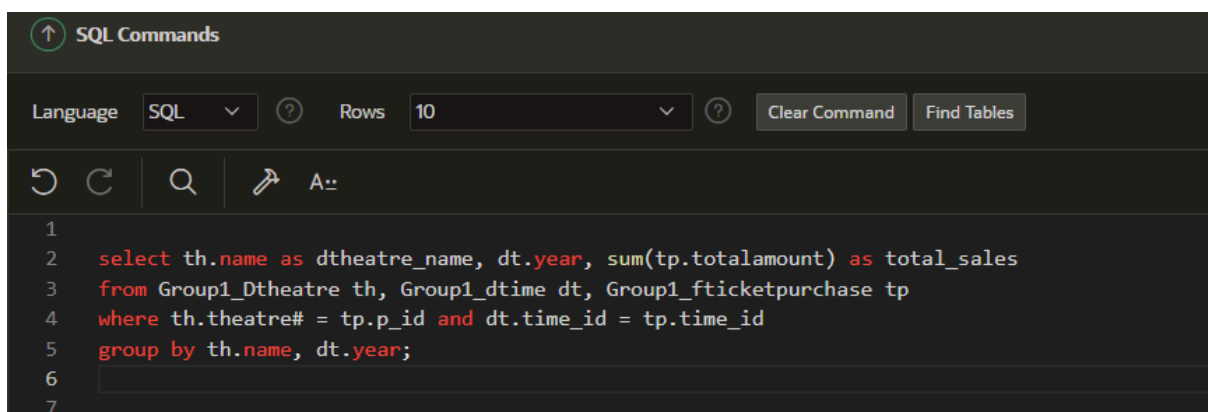


The screenshot shows a SQL interface with a dark theme. At the top, there's a header 'SQL Commands' with an upward arrow icon. Below it, a toolbar contains 'Language' set to 'SQL', a help icon, 'Rows' set to '100000', another help icon, and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a row of icons: a refresh icon, a circular arrow icon, a search icon, a pencil icon, and a text input field containing 'A..'. The main area displays a SQL query on a line-numbered background (lines 1-7):

```
1 select sum(totalamount) as "total sales", p.title
2 from ops$yyang00.ticketpurchase tp inner join ops$yyang00.production p
3 on p.p# = tp.per# group by (p.title) order by sum(totalamount) desc;
4
5
6
7
```

The theatre name and total yearly sale for each theatre :

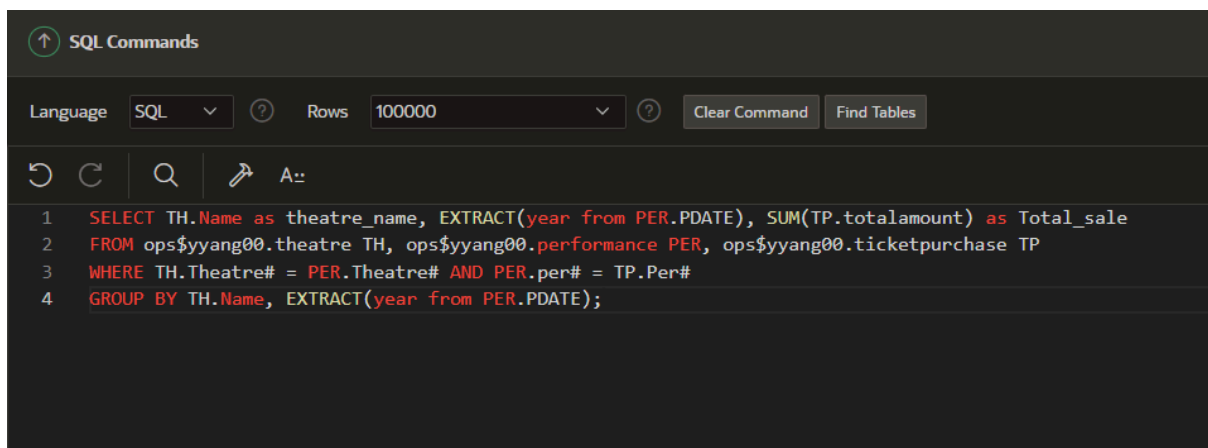
Data mart



The screenshot shows a SQL interface with a dark theme. At the top, there's a header 'SQL Commands' with an upward arrow icon. Below it, a toolbar contains 'Language' set to 'SQL', a help icon, 'Rows' set to '10', another help icon, and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a row of icons: a refresh icon, a circular arrow icon, a search icon, a pencil icon, and a text input field containing 'A..'. The main area displays a SQL query on a line-numbered background (lines 1-7):

```
1
2 select th.name as dtheatre_name, dt.year, sum(tp.totalamount) as total_sales
3 from Group1_Dtheatre th, Group1_dtime dt, Group1_fticketpurchase tp
4 where th.theatre# = tp.p_id and dt.time_id = tp.time_id
5 group by th.name, dt.year;
6
7
```

Relational model



The screenshot shows a SQL interface with a dark theme. At the top, there's a header 'SQL Commands' with an upward arrow icon. Below it, a toolbar contains 'Language' set to 'SQL', a help icon, 'Rows' set to '100000', another help icon, and buttons for 'Clear Command' and 'Find Tables'. Below the toolbar is a row of icons: a refresh icon, a circular arrow icon, a search icon, a pencil icon, and a text input field containing 'A..'. The main area displays a SQL query on a line-numbered background (lines 1-4):

```
1 SELECT TH.Name as theatre_name, EXTRACT(year from PER.PDATE), SUM(TP.totalamount) as Total_sale
2 FROM ops$yyang00.theatre TH, ops$yyang00.performance PER, ops$yyang00.ticketpurchase TP
3 WHERE TH.Theatre# = PER.Theatre# AND PER.per# = TP.Per#
4 GROUP BY TH.Name, EXTRACT(year from PER.PDATE);
```


The clients visited MT in at least 6 different months in a year

The number of clients who have only visited MT theatres in one single month and no visit in any other month :

Data mart

```
SQL Commands

Language SQL ? Rows 10 ? Clear Command Find Tables

1
2 SELECT
3     Group1_dclient.CLIENT_ID,
4     COUNT(DISTINCT Group1_dtime.MONTH) AS VMonth
5 FROM
6     Group1_fticketpurchase
7 JOIN Group1_dclient ON Group1_fticketpurchase.CLIENT_ID = Group1_dclient.CLIENT_ID
8 JOIN Group1_dtheatre ON Group1_fticketpurchase.THEATRE_ID = Group1_dtheatre.THEATRE_ID
9 JOIN Group1_dtime ON Group1_fticketpurchase.TIME_ID = Group1_dtime.TIME_ID
10 WHERE
11     Group1_dtheatre.NAME IN ('CROPSTON', 'CLINTON', 'BIRSTALL', 'VICTORY', 'BEESTON', 'VUE')
12 GROUP BY
13     Group1_dclient.CLIENT_ID
14 HAVING
15     COUNT(DISTINCT Group1_dtime.MONTH) = 1;
```

The month of the year with the highest ticket sale for a theatre each year :

Data mart

```
SQL Commands

Language SQL ? Rows 10 ? Clear Command Find Tables

1
2 SELECT
3     Group1_theatre.NAME AS TheatreName,
4     Group1_dtime.YEAR,
5     Group1_dtime.MONTH,
6     SUM(fticketpurchase.TOTALAMOUNT) AS TotalTicketSales
7 FROM
8     Group1_fticketpurchase
9 JOIN Group1_dtheatre ON Group1_fticketpurchase.THEATRE_ID = Group1_dtheatre.THEATRE_ID
10 JOIN Group1_dtime ON Group1_fticketpurchase.TIME_ID = Group1_dtime.TIME_ID
11 GROUP BY
12     Group1_dtheatre.NAME,
13     Group1_dtime.YEAR,
14     Group1_dtime.MONTH
15 ORDER BY
16     Group1_dtheatre.NAME,
17     Group1_dtime.YEAR,
18     Group1_TotalTicketSales DESC;
```

The most popular row type in each theatre by value of ticket sale :

data mart

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

↶ ↷ 🔍 ↗ A::

```
1 SELECT TheatreName,ROWTYPE,TotalTicketSales FROM
2 (SELECT dt.NAME AS TheatreName,dr.ROWTYPE, SUM(ft.TOTALAMOUNT) AS TotalTicketSales,
3 RANK() OVER (PARTITION BY ft.THEATRE_ID ORDER BY SUM(ft.TOTALAMOUNT) DESC) AS RowTypeRank FROM Group1_fticketpurchase ft
4 JOIN Group1_dthrow dr ON ft.TR_ID = dr.TR_ID
5 JOIN Group1_dtheatre dt ON ft.THEATRE_ID = dt.THEATRE_ID
6 GROUP BY
7 dt.NAME,
8 dr.ROWTYPE,
9 ft.THEATRE_ID) WHERE RowTypeRank = 1
10 ORDER BY TheatreName;
```

5.2 Evidence of the SQL code working and their results

Production-wise reporting of daily analysis :

Data mart

Results	Explain	Describe	Saved SQL	History
FAMILY		5	422.5	
IMPOSSIBLE		4	6875	
MORNING SUNSHINE		4	455	
FAMILY		11	572.5	
VILLAGES		3	125	
WINDOWS		9	60	
STAR WAR		8	32.5	
BLUE SKY		25	120	
PINK FLIGHT		29	62.5	
NEW WORLD		15	30	
YOUR TURN		19	1575	
OLD MAN		19	30	
2228 rows returned in 0.07 seconds Download				

Relational model

Results	Explain	Describe	Saved SQL	History
Impossible		18	100	
Strike		13	30.5	
Strike		18	107	
Story of Kitchen		21	69.5	
Story of Kitchen		25	42.5	
Morning Sunshine		17	51	
London Bridge		14	42.5	
London Bridge		17	56	
Berlin Wall		31	56	
Strike		23	50.5	
Morning Sunshine		23	12.5	
Once Again		30	75.5	
2261 rows returned in 0.07 seconds Download				

The title and total sale of each production :

Data mart

Results	Explain	Describe	Saved SQL	History
9110				THE WAY TO SUCCESS
9087.5				SMOKE
9025				VILLAGES
9002.5				NEIGHBOURHOOD
8537.5				BEACH
8335				WINTER WIND
8257.5				STRIKE
7752.5				IMPOSSIBLE
7620				YOUR TURN
5485				BACK TO HOME
5030				FAMILY
4365				LASTING FRIENDSHIP

Relational model

Results	Explain	Describe	Saved SQL	History
19				Panda
18.5				Pacific
18.5				Two Sisters
18.5				Flight to Paris
18.5				The World
18				The Way to Success
12.5				The Late Spring
12				Snow Man
12				Windows
6.5				Cloud
6				Once Agin
77 rows returned in 0.01 seconds Download				

The theatre name and total yearly sale for each theatre :

Data mart

Results

Explain

Describe


Saved SQL


History

DTHEATRE_NAME		YEAR	TOTAL_SALES
VUE		2022	49275
VICTORY		2022	49152.5
BEESTON		2022	10102.5
CROPSTON		2022	11475
CLINTON		2023	10645
BIRSTALL		2022	11922.5

6 rows returned in 0.07 seconds

Download

p2797471

p2797471

en

Copyright © 1999, 2025, Oracle and/or its affiliates.




Oracle APEX 25.1.2

Relational model

Results	Explain	Describe	Saved SQL	History
victory			2025	11617.5
Clinton			2023	10192.5
Cropston			2023	15597
Vue			2022	14161.5
Clinton			2022	13236.5
Cropston			2022	21490.5
victory			2022	15480.5
Birstall			2023	15059.5
beeston			2023	10949.5
12 rows returned in 0.01 seconds Download				
p2797471 p2797471 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.2				

The month of the year with the highest ticket sale for a theatre each year :

Data mart

Results	Explain	Describe	Saved SQL	History	
VUE			2022	7	5972.5
VUE			2022	8	5615
VUE			2022	10	5582.5
VUE			2023	6	8785
VUE			2023	9	7595
VUE			2023	3	7595
VUE			2023	7	6912.5
VUE			2023	5	6077.5
VUE			2023	4	5305
VUE			2023	8	5095
VUE			2023	10	4695
62 rows returned in 0.02 seconds Download					
<div> p2810257  p2810257  en</div> <div>Copyright © 1999, 2023, Oracle and/or its affiliates</div> <div>Oracle APEX 23.1.2</div>					

The number of clients who have only visited MT theatres in one single month and no visit in any other month :

Data mart

Results	Explain	Describe	Saved SQL	History
1098				1
175				1
2377				1
4420				1
3723				1
1163				1
2447				1
2628				1
1227				1
634 rows returned in 0.03 seconds Download				
p2810257 p2810257 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.2				

The most popular row type in each theatre by value of ticket sale :

Data mart

Results	Explain	Describe	Saved SQL	History
THEATRENAME		ROWTYPE		TOTALTICKETSALES
BEESTON		REAR		48182.5
BIRSTALL		REAR		67615
CLINTON		FRONT		44332.5
CROPSTON		FRONT		70360
VICTORY		REAR		49812.5
Vue		REAR		46635
6 rows returned in 0.02 seconds Download				
p2810257 p2810257 en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.2				

6. Justification of the data mart design and comparison of data mart and OLTP

6.1 Please compare your solutions in 5 and comment on how they satisfy the requirements of the data analysis. Furthermore, please compare Data Mart and the relational model in data analysis in general.

Justification of the Data Mart Design

In a data-driven world, many organizations are faced with a large amount of data. From employee records to customer information, which is why there is a need for data management and analysis.

A data mart is a data warehouse system that contains specific data in an organization's unit. It usually contains a small, selected part of data that a company will store in a bigger data pool. Companies use data marts to be able to run analyses of specific information to operate more effectively.

Firstly, by organising relevant datasets into subgroups, this allows for more straightforward navigation and understanding of complex data structures. In the star schema 1.1 the data sets are centralised by the Fact table of Ticket sales. The fact table represents the central entity in the database, which contains measurable and numeric data related to a specific event or process. To be able to identify each row within the fact table, the primary keys are established in the star schema design. In the star schema design 1.1 it is always beneficial to assign unique identifiers to the fact such as P_ID, TR_ID, Client_ID, Theatre_ID and Time_ID as foreign keys to maintain accuracy and avoid redundancy errors. By doing so, this reduces computational overload by limiting calculations on insignificant dimensions not involved in the query.

The foreign keys create the correlation between multiple tables within the database by referencing the primary keys from other sub-group tables. This link helps data remain organised and links various dimensions and their facts. In ticket sales, the foreign keys can connect information such as the production and theatre with their respective time records stored in another dimension table.

Comparison of data mart and OLTP

The relational model and the data mart are two models that are frequently used in data analysis. Both have their qualities and benefits, but they both have the same function of gathering and examining data to produce insightful information.

One conventional method that is frequently used to arrange vast amounts of structured data is the relational model. It represents entities and their relationships using tables with rows and columns. Primary keys and foreign keys define the relationships between the entities.

A data mart is a portion of a company's data warehouse that is dedicated to divisions inside the company. It is intended to facilitate decision-making by making data easily accessible for analysis. A data mart is more focused on topic areas like sales or finance than the relational model, which includes multiple entities across a whole organisation.

Both models are functional, allowing users to query and analyse datasets. Structured Query Language (SQL) is used to write queries in relational database systems that obtain needed data from many databases based on specified relationships. This makes it possible to run queries that involve table joins.

For our case, using a data mart is the best solution. Analysing business performance becomes a smoother process. The data mart design responds to user needs, ensuring meaningful insights for informed business decisions regarding MT performance. Overall, while OLTP and Data Mart contribute distinctly to data analysis, the latter stands out for its focused approach, simplifying complex queries for strategic decision-making.

7. Critical analysis and reflection

7.1 Please give a critical analysis on your group's work, what is your success, what is not as expected. How is your group collaboration. With more time, what could be further improved. The management evaluation report from each member will have impact on this part.

Our data warehouse project has been both highly enriching and instructive. Our group has had some major successes but also some difficulties.

The first step was the realization of the dimensions tables and the fact one in our data warehouse model. Collaboration within the group was essential to discussing different ideas or help those who needed it. We discussed together to make one important step: who has the determination of tables and attributes would be most useful.

However, we had a few problems with this part. This part was the most important, as it would guide the rest of our work. At first, we wanted to use the 'Performance' table as a fact table. However, we took a closer look at the needs of the data mart and came to the conclusion that the 'TicketPurchase' table was the most appropriate because, according to the needs, the key factor was the analysis of ticket sales. So, for our purposes, the 'TicketPurchase' table was the best, thanks in particular to the 'TotalAmount' attribute.

This part helped us to understand the user requirements for our case study. The fair distribution of tasks also boosted our efficiency, with everyone contributing their skills to the project.

However, we also encountered difficulties in setting up the ETL. This phase of the project required us to think carefully about data transformation and loading, and we had to overcome a number of technical obstacles. This difficulty turned out to be an opportunity to strengthen our understanding of the subject.

Initially, we used all the table attributes we wanted to use. Over time, we realized that this was a bad idea. We then completely revised our use of attributes for each table. After re-reading all the tables, we used only some of the most important attributes. By dedicating more resources to this stage, we were able to optimize the relevance of the data stored and improve the overall quality of our data warehouse. Collaboration within the group was a strong point, but with more time, we could have organized additional full working sessions to improve our understanding of the technical aspects of the project.

When we were creating the ETL query for the 'group1_FITCKETPURCHASE' fact table, we encountered a problem with the insert part. Thanks to our tests, we realized that our query was correct, but the insertion didn't work, and we got a message telling us that our storage space in our ORACLE Apex session was full. After that, we ran several tests on different computers without being able to solve the problem.

We then looked at the number of rows in the 'TICKETPURCHASE' table to determine the number of rows to insert in the 'group1_FITCKETPURCHASE' table. This test returned 27868 rows. We then used the 'FETCH FIRST 27868 ROWS ONLY' command, which solved our insertion problem. However, some data may not have been inserted, as there may be more than 27868 rows of data.

In addition, we had some difficulty realizing the queries in part 5. We worked together to realize them, but some difficulties were present. Additional time would have been beneficial to finalize the last queries. A lack of time management may have been detrimental to this part. Changes to the choice of data marts in the middle of the project certainly had a negative impact.

In brief, despite the challenges, completing this project reinforced our understanding of key data warehousing concepts and underlined the importance of collaboration and planning. We were quickly able to create a group and start working on the tasks, as well as dividing up the work in our group. This gave us more time to think about the questions posed and how to answer them. With more time, we could have perfected some aspects of the project, but the experience we gained provides a solid basis for future projects.

Management Evaluation Report

Student ID	P2739759	Contribution %	12.5
Justification	I worked on those task in this report: 1.1, 1.2(b), 3.1, 3.2, 4.2, 4.3		

Student ID	P2810257	Contribution %	12.5
Justification	I have worked on the whole project collaboratively, But mainly on the analysis part task 5/5.1 /5.2. present at all organized meetings		

Student ID	P2797471	Contribution %	12.5
Justification	I worked on the whole project, but more particularly on part 5/51/5.1/5.2. Present at all organized meetings.		

Student ID	P2739646	Contribution %	12.5
Justification	I worked on the whole project, but more particularly on part 1.1/1.2-c/3/4/7. Present at all organized meetings.		

Student ID	P2600762	Contribution %	12.5
Justification	Throughout the assignment, I ensured consistent dedication throughout the project. My main contribution was creating all the tables which were in use throughout the report, assigning the correct data types as well as all the appropriate labels.		

Student ID	P2809461	Contribution %	12.5
Justification	I worked on the whole project, particularly for 1.1/1.2/2.1/2.2 Present for all meetings		

Student ID	P2777638	Contribution %	12.5
Justification	I worked on part 3		

Student ID	P17155768	Contribution %	12.5
------------	-----------	----------------	------

Justification	Throughout the project, I have maintained consistency within the work taking up the justification of data analysis and the comparison of the relational model and OLTP while engaging with group members to ensure we have completed each task appropriately.
---------------	---

Appendix

--for group1_dproduction

```
CREATE TABLE group1_DPRODUCTION (
P_ID NUMBER(5) PRIMARY KEY,
P# NUMBER (5) NOT NULL,
TITLE VARCHAR2(20) NOT NULL);
```

```
CREATE SEQUENCE DP_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
```

```
INSERT INTO group1_DPRODUCTION
SELECT DP_seq.nextval, P#, TITLE
FROM (SELECT DISTINCT P#, UPPER (TRIM(TITLE)) TITLE
FROM (SELECT P.P#, P.TITLE
FROM ops$yyang00.production p,
ops$yyang00.PERFORMANCE PER,
ops$yyang00.TICKETPURCHASE TP
WHERE P.P# = PER.P# AND TP.PER# = PER.PER#)) );
```

--FOR group1_DTROW

```
create table group1_DTROW (
TR_ID NUMBER(5) PRIMARY KEY,
TR# NUMBER(5) NOT NULL,
ROWTYPE VARCHAR2(10),
THEATRE# NUMBER(5))
```

```
CREATE SEQUENCE DTR_seq
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
```

```
INSERT INTO group1_DTROW
SELECT DTR_seq.nextval, TR#, ROWTYPE, THEATRE#
FROM (SELECT DISTINCT UPPER (TRIM(ROWTYPE)) ROWTYPE, TR#, THEATRE#
FROM (SELECT tr.TR#, tr.ROWTYPE, TR.THEATRE# FROM ops$yyang00.TROW TR, ops$yyang00.TSEAT TS
WHERE TR.TR# = TS.TR#));
```

--FOR group1_DTHEATRE

```
SELECT * FROM ops$yyang00.THEATRE
```

```
CREATE TABLE group1_DTHEATRE (
THEATRE_ID NUMBER (5) PRIMARY KEY,
```



```
THEATRE# NUMBER (5) NOT NULL,  
NAME VARCHAR2(20) NOT NULL);
```

```
CREATE SEQUENCE DTHEATRE_seq  
START WITH 1  
INCREMENT BY 1  
NOCACHE  
NOCYCLE;
```

```
INSERT INTO group1_DTHEATRE  
SELECT DTHEATRE_seq.nextval, THEATRE#, NAME  
FROM (SELECT DISTINCT THEATRE#, UPPER (TRIM(NAME)) NAME  
FROM (SELECT T.THEATRE#, T.NAME  
FROM ops$yyang00.THEATRE T,  
ops$yyang00.PERFORMANCE PER,  
ops$yyang00.TICKETPURCHASE TP  
WHERE T.THEATRE# = PER.THEATRE# AND TP.PER# = PER.PER#));
```

```
--FOR group1_DCLIENT
```

```
CREATE TABLE group1_DCLIENT (  
CLIENT_ID NUMBER (5) PRIMARY KEY,  
CLIENT# NUMBER(5) NOT NULL,  
TITLE VARCHAR2(20) NOT NULL,  
NAME VARCHAR2(20) NOT NULL);
```

```
CREATE SEQUENCE DCLIENT_seq  
START WITH 1  
INCREMENT BY 1  
NOCACHE  
NOCYCLE;
```

```
INSERT INTO group1_DCLIENT  
SELECT DCLIENT_seq.nextval, CLIENT#, TITLE, NAME  
FROM (SELECT DISTINCT CLIENT#, UPPER (TRIM(TITLE)) TITLE, UPPER(TRIM(NAME)) NAME  
FROM (SELECT C.CLIENT#, C.TITLE, C.NAME FROM ops$yyang00.CLIENT C, ops$yyang00.TICKETPURCHASE TP  
WHERE C.CLIENT# = TP.CLIENT#));
```

```
--FOR group1_DTIME
```

```
CREATE TABLE group1_DTIME (  
TIME_ID NUMBER (5) PRIMARY KEY,  
YEAR number(4) NOT NULL,  
MONTH NUMBER(2) NOT NULL,  
DAY NUMBER(2) NOT NULL);
```

```
CREATE SEQUENCE DTIME_seq  
START WITH 1  
INCREMENT BY 1  
NOCACHE  
NOCYCLE;
```

```
INSERT INTO group1_DTIME  
SELECT DTIME_seq.nextval, YEAR, MONTH, DAY  
FROM (SELECT DISTINCT  
EXTRACT (Year FROM PDATE) Year,  
EXTRACT (Month FROM PDATE) Month,
```

```
EXTRACT (DAY FROM PDATE) DAY
FROM ops$yyang00.PERFORMANCE);
```

```
--FOR group1_FTICKETPURCHASE
```

```
CREATE TABLE group1_FTICKETPURCHASE (
THEATRE_ID NUMBER(5) CONSTRAINT fk11 REFERENCES group1_DTHEATRE,
P_ID NUMBER(5) CONSTRAINT fk12 REFERENCES group1_DPRODUCTION,
TR_ID NUMBER(5) CONSTRAINT FK13 REFERENCES group1_DTROW,
CLIENT_ID NUMBER(5) CONSTRAINT fk14 REFERENCES group1_DCLIENT,
TIME_ID NUMBER(5) CONSTRAINT fk15 REFERENCES group1_DTIME,
TOTALAMOUNT NUMBER(10,2) NOT NULL,
CONSTRAINT pk11 PRIMARY KEY (THEATRE_ID, P_ID, TR_ID, CLIENT_ID, TIME_ID));
```

```
INSERT INTO group1_FTICKETPURCHASE
SELECT THEATRE_ID, P_ID, TR_ID, CLIENT_ID, TIME_ID, TOTALAMOUNT
FROM (
  SELECT
    T.THEATRE_ID, P.P_ID, C.CLIENT_ID, TM.TIME_ID, TR.TR_ID, SUM(TP.TOTALAMOUNT) AS TOTALAMOUNT
  FROM
    ops$yyang00.PERFORMANCE PER,
    group1_DCLIENT C,
    group1_DTHEATRE T,
    group1_DPRODUCTION P,
    group1_DTIME TM,
    group1_DTROW TR,
    ops$yyang00.TICKETPURCHASE TP,
    ops$yyang00.TSEAT TS
  WHERE
    PER.THEATRE# = T.THEATRE#
    AND PER.P# = P.P#
    AND TS.TR# = TR.TR#
    AND TP.CLIENT# = C.CLIENT#
    AND TP.PER# = PER.PER#
    AND EXTRACT(YEAR FROM PER.PDATE) = TM.YEAR
    AND EXTRACT(MONTH FROM PER.PDATE) = TM.MONTH
    AND EXTRACT(DAY FROM PER.PDATE) = TM.DAY
  GROUP BY
    T.THEATRE_ID, P.P_ID, TR.TR_ID, C.CLIENT_ID, TM.TIME_ID
  FETCH FIRST 27868 ROWS ONLY)
```