Student ID: 34389377 Ritika Raina

FIT3003: Major Assignment 2

Monash Equipment Centre: Dataware House and Analysis

Table of Contents

FIT3003: Major Assignment 2 **1**

Table of Contents 2

A. Transformation Stage **3**

(a) Preparation Stage 3

(b) Design Task A 10

(c) Suggestion to increase the granularity 10

(d) SCD 11

Star Schema A implementation **12**

B. Analysis Stage **14**

1. Equipments Sold vs Hired 14
2. Revenue generation through the years (2018-2020) 15
3. Product Category Analysis 17

A. Transformation Stage

(a) Preparation Stage

Error 1 & 2: A null value as description in the category table, also affecting the equipment table

BEFORE

Code:

```
SELECT *
FROM ME_CATEGORY;
```

Output:

	CATEGORY_ID		CATEGORY_DESCRIPTION
4		4	Concrete
5		5	Earthmoving
6		6	Generators
7		7	Landscaping
8		8	Lighting
9		9	Plumbing
10		10	Rail
11		11	Safety
12		12	Site Equipment
13		13	Trailers
14		14	Vehicles
15		15	null

But before removing this, we need to check the other tables where a foreign key has been used a foreign key.

Code:

```
SELECT *
FROM ME_EQUIPMENT
WHERE CATEGORY_ID = 15;
```

Category_id is also used in the equipment table. Over here we can see that the equipment table has a record which has the category id 15.

Output:

	EQUIPMENT_ID	EQUIPMENT_NAME	EQUIPMENT_PRICE	MANUFACTURE_YEAR	MANUFACTURER	CATEGORY_ID
1	158	EXCAVATOR - POST HOLE ATTACHMENT SUIT 3.5T	12200	2017	HITACHI	15

The equipment_id 158 has an invalid category_id. But seeing that the other tables that use equipment_id don't have a value of equipment id 158. So we don't need to worry about it.
So we delete this record from the equipment table and the category table.

AFTER

Code:

```
DELETE FROM ME_CATEGORY
WHERE CATEGORY_ID = 15;
```

Output:

	CATEGORY_ID	CATEGORY_DESCRIPTION
3	3	Compaction
4	4	Concrete
5	5	Earthmoving
6	6	Generators
7	7	Landscaping
8	8	Lighting
9	9	Plumbing
10	10	Rail
11	11	Safety
12	12	Site Equipment
13	13	Trailers
14	14	Vehicles

Now only 14 rows are left in the category table.

Code:

```
DELETE FROM ME_EQUIPMENT
WHERE CATEGORY_ID = 15;
```

Output:

The record from the equipment table which had category id 15 is removed as well.

```
-- Viewing the table as whole
SELECT *
FROM ME_EQUIPMENT
WHERE CATEGORY_ID = 15;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 0 in 0.148 seconds

	EQUIPMENT_ID	EQUIPMENT_NAME	EQUIPMENT_PRICE	MANUFACTURE_YEAR	MANUFACTURER	CATEGORY_ID

Error 3: Customer table duplicates

BEFORE

Code:

```
SELECT *
FROM ME_CUSTOMER
WHERE (CUSTOMER_ID, CUSTOMER_TYPE_ID, NAME, GENDER, ADDRESS_ID, PHONE,
EMAIL) IN (
    SELECT CUSTOMER_ID, CUSTOMER_TYPE_ID, NAME, GENDER, ADDRESS_ID,
PHONE, EMAIL
    FROM ME_CUSTOMER
```

```
    GROUP BY CUSTOMER_ID, CUSTOMER_TYPE_ID, NAME, GENDER, ADDRESS_ID,
PHONE, EMAIL
    HAVING COUNT(*) > 1
)
ORDER BY CUSTOMER_ID, NAME;
```

Output:

	CUSTOMER_ID	CUSTOMER_TYPE_ID	NAME	GENDER	ADDRESS_ID	PHONE	EMAIL
1	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu
2	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu
3	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu
4	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu

AFTER

Code:

```
DELETE FROM ME_CUSTOMER
WHERE ROWID NOT IN (
    SELECT MIN(ROWID)
    FROM ME_CUSTOMER
    GROUP BY CUSTOMER_ID
);
```

```
SELECT COUNT(*) - COUNT(DISTINCT CUSTOMER_ID || '-' || CUSTOMER_TYPE_ID
|| '-' || NAME|| '-' || GENDER || '-' || ADDRESS_ID || '-' ||PHONE||
'-' ||EMAIL ) AS DUPLICATES
FROM ME_CUSTOMER;
```

Output:

We remove the duplicates

All rows fetched: 1 in 0.190 seconds

	DUPLICATES
1	0

Error 4: Customer table gender values are inconsistent

BEFORE

Code:

```
SELECT *
FROM ME_CUSTOMER
WHERE GENDER <> 'Male'
  AND GENDER <> 'Female';
```

Output:

	CUSTOMER_ID	CUSTOMER_TYPE_ID	NAME	GENDER	ADDRESS_ID	PHONE	EMAIL
23	23	1	Astra Petlyura	Female	23	410 351 7284	apetlyuram@bizjournals.com
24	24	2	Enrika Aleksich	Female	24	731 262 1586	ealeksichn@ameblo.jp
25	25	2	Burton Allery	Male	25	610 880 4187	balleryo@pbs.org
26	26	1	Germayne Winsper	Male	26	301 243 7927	gwinsperp@phoca.cz
27	27	2	Darrelle Durgan	F	27	659 308 2928	ddurganq@mapquest.com
28	28	2	Ruttger Hamfleet	Male	28	313 240 5119	rhamfleetr@howstuffworks.com
29	29	1	West Kopf	Male	29	786 152 7034	wkopfs@umich.edu
30	30	1	Adlai Dunbavin	Male	30	413 346 6507	adunbavint@bluehost.com
31	31	1	Patti Creggan	Female	31	289 284 5903	pcregganu@prnewswire.com
32	32	1	Grata Newcom	Female	32	996 106 1417	gnewcomv@blogtalkradio.com

```
SELECT *
FROM ME_CUSTOMER
WHERE GENDER <> 'Male'
  AND GENDER <> 'Female';
------------------------------------- customer_type TABLE
-- Creating a duplicate table
Create table ME_CUSTOMERTYPE as
Select *
From MonEquip.Customer_Type;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 4 in 0.149 seconds

	CUSTOMER_ID	CUSTOMER_TYPE_ID	NAME	GENDER	ADDRESS_ID	PHONE	EMAIL
1	27	2	Darrelle Durgan	F	27	659 308 2928	ddurganq@mapquest.com
2	79	1	Swen Huddlestone	M	79	362 978 1451	shuddlestone26@webnode.com
3	87	2	Wilma Abyss	F	87	888 141 6586	wabyss2e@xinhuanet.com
4	139	1	Ned Halliberton	M	139	459 345 6431	nhalliberton3u@com.com

AFTER

Code:

```
UPDATE ME_CUSTOMER
SET GENDER = 'Male'
WHERE GENDER IN ('M');

UPDATE ME_CUSTOMER
SET GENDER = 'Female'
WHERE GENDER IN ('F');
```

Output:

We will replace these errors with the full form for F and M.

All rows fetched: 4 in 0.163 seconds

	CUSTOMER_ID	CUSTOMER_TYPE_ID	NAME	GENDER	ADDRESS_ID	PHONE	EMAIL
1	27	2	Darrelle Durgan	Female	27	659 308 2928	ddurganq@mapquest.com
2	79	1	Swen Huddlestone	Male	79	362 978 1451	shuddlestone26@webnode.com
3	87	2	Wilma Abyss	Female	87	888 141 6586	wabyss2e@xinhuanet.com
4	139	1	Ned Halliberton	Male	139	459 345 6431	nhalliberton3u@com.com

The customer ids where the gender values were inconsistent, have been updated.

Error 5: Duplicate in Customer type table

BEFORE

Code:

```
SELECT *
FROM ME_CUSTOMERTYPE;
```

Output:

	CUSTOMER_TYPE_ID	DESCRIPTION
1	1	Individual
2	2	Business
3	2	business

Both these records are stating the same thing, so we will just delete the record with the description of 'business'.

AFTER

Code:

```
DELETE FROM ME_CUSTOMERTYPE
WHERE description = 'business';
```

Output:

	CUSTOMER_TYPE_ID	DESCRIPTION
1	1	Individual
2	2	Business

Error 6: Invalid date spotted in the Hire table

BEFORE

Code:

```
SELECT hire_id, START_DATE, END_DATE, END_DATE - START_DATE AS
duration_days
FROM ME_HIRE
WHERE END_DATE-START_DATE <0;
```

Output:

	HIRE_ID	START_DATE	END_DATE	DURATION_DAYS
1	302	05/12/20	17/10/20	-49

Hire id 302 has a start date after the end date, we can see from the negative duration days calculated.
So we delete this record.

AFTER

Code:

```
DELETE FROM ME_HIRE
WHERE HIRE_ID = 302;
```

Output:

No output after we search for invalid dates.

```
-- Checking for any invalid dates
SELECT hire_id, START_DATE, END_DATE, END_DATE - START_DATE AS duration_days
FROM ME_HIRE
WHERE END_DATE-START_DATE <0;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 0 in 0.125 seconds

HIRE_ID	START_DATE	END_DATE	DURATION_DAYS

Error 7: Invalid staff id spotted in the Hire table

BEFORE

Code:

```
select * from me_hire where staff_id >51;
```

Output:

	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID
1	301	08/12/2020 12:00:00	08/12/2020 12:00:00	190	1	300	300	181	174
2	303	25/01/2090 12:00:00	27/12/2099 12:00:00	43	3	50	-150	53	223
3	304	08/12/2020 12:00:00	08/12/2020 12:00:00	114	1	350	-1	34	85

AFTER

Code:

```
delete from me_hire where staff_id > 51;
select * from me_hire where staff_id >51;
```

Output:

All rows fetched: 0 in 0.187 seconds

HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID

Error 8: Sales table has a negative quantity

BEFORE

Code:

```
select *
from ME_SALES
```

```
WHERE QUANTITY < 0;
```

Output:

All rows fetched: 1 in 0.114 seconds

	SALES_ID	SALES_DATE	EQUIPMENT_ID	QUANTITY	UNIT_SALES_PRICE	TOTAL_SALES_PRICE	CUSTOMER_ID	STAFF_ID
1	151	15/12/20	20	-3	45500	182000	2	37

The quantity for sales_id 151 is negative which is wrong.
So will be deleting this record as well.

AFTER

Code:

```
DELETE FROM ME_SALES
WHERE SALES_ID = 151;
select *
from ME_SALES
WHERE QUANTITY < 0;
```

Output:

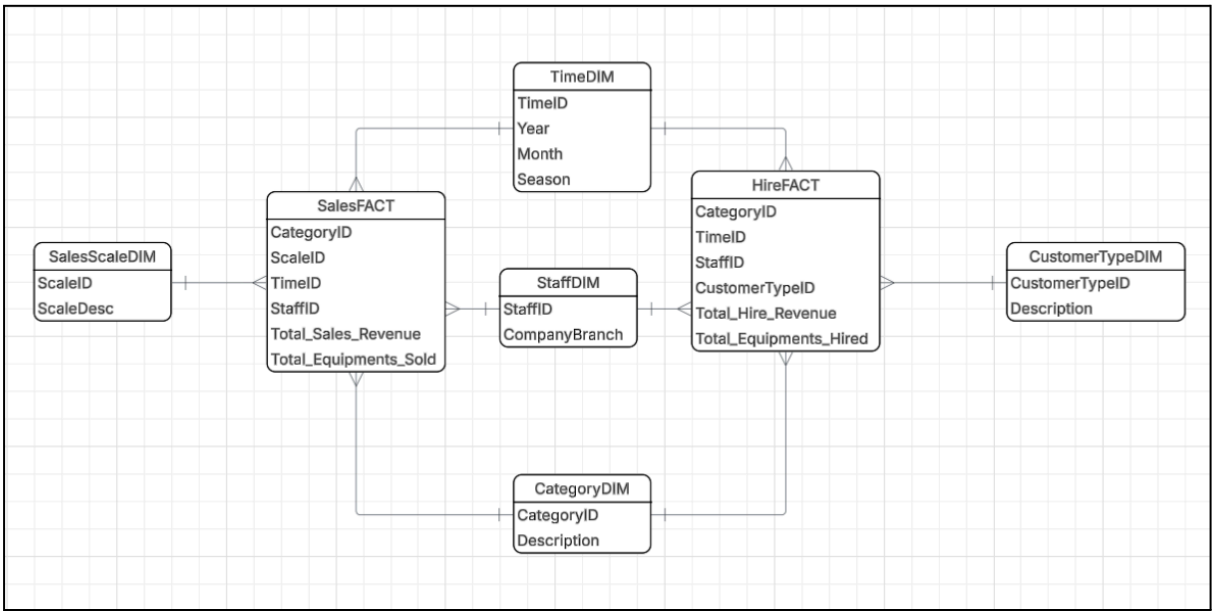
```
271  -- -ve value
272  select *
273  from ME_SALES
274  WHERE QUANTITY < 0;
275
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 0 in 0.113 seconds

SALES_ID	SALES_DATE	EQUIPMENT_ID	QUANTITY	UNIT_SALES_PRICE	TOTAL_SALES_PRICE	CUSTOMER_ID	STAFF_ID

(b) Design Task A



(c) Suggestion to increase the granularity

To increase the granularity of Star Schema A, here are a few suggestions:

1. *Adding equipment-level details*
   By introducing an Equipment Dimension, we can include the EquipmentID in the fact tables. This allows the fact tables to record exactly which equipment item was sold or hired, instead of keeping it only at the category level. It would also let us view details such as the manufacturer and the year each equipment was made.

2. *Adding a customer dimension*
   Including a Customer Dimension would help identify repeat customers and analyze customer loyalty. It would also provide more insights into how different factors, such as gender or customer type, influence the sales or hire of equipment.

3. *Expanding the staff dimension*
   By adding more details to the Staff Dimension, we can see how many staff members work in each branch and analyze the ratio of male to female employees. This can also support more detailed performance comparisons across branches.

(d) SCD

Slowly changing dimensions are dimensions which have records which change slowly over a period of time.

There are 6 types of SCD:
Type 0
Type 1: In this type of dimension, the record is changed directly and no past history is saved
Type 2: This type preserves the full history of data by creating a new record every time a record is updated. Each version of this record is assigned a unique identifier like an id which helps us track the evolution of the record.
Type 3: This type only records the current and the previous record.
Type 4: This type makes a new table which stores the complete history of each record
Type 6: This is a hybrid approach where types 1, 2 and 3 are combined together. It preserves full history with new records, tracks previous values in columns and also allows certain attributes to be updated directly for convenience.

My star schema contains a temporal dimension which is the staff dimension. Staff relocation is a very common thing, so in this dimension there can be the SCD like the following -

For easier explanation lets take for example a staff member with staff id - 1, who is currently in the Caulfield branch and is relocated to the Toorak branch.
In this case these will how the different types of SCD work:

- **Type 1:** The company branch column for staff id 1 will be directly updated
- **Type 2 :** In this case, a new record for the staff id will be created and keeping everything else the same, only the company branch for that new record will be changed to Toorak.
- **Type 3 :** A new column will be created in the dimension named something like *previous_branch*. This will store the past branch and the existing branch column would store the current branch information.
- **Type 4 :** The Staff Dimension will only keep the current branch of each staff member. To keep track of any branch changes, another dimension called StaffHistoryDIM will be added. This table will store the previous branches where the staff have worked, so we can still see their past branch information when needed.
- **Type 6 :** Both the current and previous branch details will be kept in the same record, and a new row will be added whenever a staff member moves to a new branch. This way, we can see their full branch history while still keeping their latest branch information updated.

Star Schema A implementation

Time Dimension

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	TIME_ID	NUMBER	Y	(null)	1
2	YEAR	NUMBER	Y	(null)	2
3	MONTH	NUMBER	Y	(null)	3
4	SEASON	CHAR	Y	(null)	4

Table example

	TIME_ID	YEAR	MONTH	SEASON
1	1	2018	4	Autumn
2	2	2018	4	Autumn
3	3	2018	5	Autumn
4	4	2018	5	Autumn
5	5	2018	5	Autumn
6	6	2018	5	Autumn
7	7	2018	5	Autumn
8	8	2018	5	Autumn

Staff Dimension

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	STAFF_ID	NUMBER	Y	(null)	1
2	COMPANY_BRANCH	VARCHAR2	Y	(null)	2

Table example

	STAFF_ID	COMPANY_BRANCH
1	1	Caulfield
2	2	Hughesdale
3	3	Clayton
4	4	Toorak
5	5	Clayton
6	6	Eltham
7	7	Chadstone
8	8	Docklands

Category Dimension

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	CATEGORY_ID	NUMBER	Y	(null)	1
2	CATEGORY_DESCRIPTION	VARCHAR2	Y	(null)	2

Table example

	CATEGORY_ID	CATEGORY_DESCRIPTION
1	1	Access
2	2	Air Compressor
3	3	Compaction
4	4	Concrete
5	5	Earthmoving
6	6	Generators
7	7	Landscaping
8	8	Lighting

CustomerType Dimension

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	CUSTOMERTYPEID	NUMBER	Y	(null)	1
2	DESCRIPTION	VARCHAR2	Y	(null)	2

Table example

	CUSTOMERTYPEID	DESCRIPTION
1	1	Individual
2	2	Business

SalesScale Dimension

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	SCALEID	NUMBER	N	(null)	1
2	SCALEDESC	VARCHAR2	Y	(null)	2

Table example

	SCALEID	SCALEDESC
1	1	Low Sales (Below $5000)
2	2	Medium Sales ($5000 - $10000)
3	3	High Sales (Above $10000)

Hire Fact Table

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	CATEGORY_ID	NUMBER	Y	(null)	1
2	STAFF_ID	NUMBER	Y	(null)	2
3	TIME_ID	NUMBER	Y	(null)	3
4	CUSTOMER_TYPE_ID	NUMBER	Y	(null)	4
5	TOTAL_HIRE_REVENUE	NUMBER	Y	(null)	5
6	TOTAL_EQUIPMENTS_HIRED	NUMBER	Y	(null)	6

Table example

	CATEGORY_ID	STAFF_ID	TIME_ID	CUSTOMER_TYPE_ID	TOTAL_HIRE_REVENUE	TOTAL_EQUIPMENTS_HIRED
1	12	2	3	1	240	3
2	12	2	10	1	240	3
3	12	2	12	1	240	3
4	10	17	6	2	150	1
5	3	35	4	2	1200	2
6	5	31	9	1	1000	2
7	7	10	3	1	420	3
8	7	10	10	1	420	3

Sales Fact Table

Table Structure

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	CATEGORY_ID	NUMBER	Y	(null)	1
2	STAFF_ID	NUMBER	Y	(null)	2
3	TIME_ID	NUMBER	Y	(null)	3
4	SCALE_ID	NUMBER	Y	(null)	4
5	TOTAL_SALES_REVENUE	NUMBER	Y	(null)	5
6	TOTAL_EQUIPMENTS_SOLD	NUMBER	Y	(null)	6

Table example

	CATEGORY_ID	STAFF_ID	TIME_ID	SCALE_ID	TOTAL_SALES_REVENUE	TOTAL_EQUIPMENTS_SOLD
1	7	39	10	3	83200	2
2	13	16	13	3	18000	2
3	5	37	20	3	216000	4
4	11	26	20	3	11200	2
5	5	37	21	3	216000	4
6	3	22	22	3	60000	3
7	5	37	22	3	216000	4
8	11	26	22	3	11200	2

B. Analysis Stage

1. *Equipments Sold vs Hired*

Code:

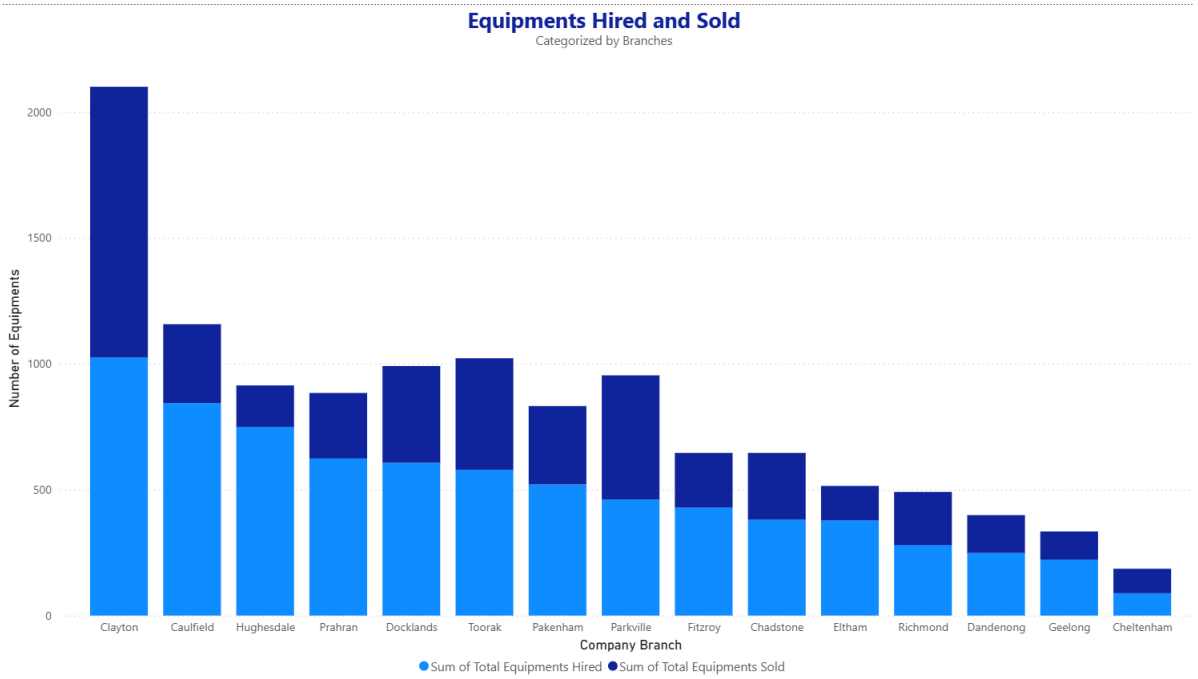
```
SELECT
    s.company_branch,
    NVL(SUM(sf.total_sold), 0) AS total_sold,
    NVL(SUM(hf.total_hired), 0) AS total_hired
FROM StaffDIM s
LEFT JOIN (
    SELECT staff_id, SUM(total_equipments_sold) AS total_sold
    FROM SalesFact
    GROUP BY staff_id
```

```
) sf ON s.staff_id = sf.staff_id
LEFT JOIN (
    SELECT staff_id, SUM(total_equipments_hired) AS total_hired
    FROM HireFact
    GROUP BY staff_id
) hf ON s.staff_id = hf.staff_id
GROUP BY s.company_branch;
```

Output:

	COMPANY_BRANCH	TOTAL_SOLD	TOTAL_HIRED
1	Richmond	211	280
2	Pakenham	311	521
3	Caulfield	313	844
4	Clayton	1075	1025
5	Docklands	384	607
6	Parkville	493	461
7	Dandenong	150	249
8	Eltham	137	378
9	Toorak	443	579
10	Chadstone	265	381
11	Geelong	112	222
12	Prahran	260	624
13	Cheltenham	97	89
14	Hughesdale	165	749
15	Fitzroy	217	429

Visualisation:



The stacked bar chart created in PowerBI, representing the code output from the OLAP query, presents the number of equipment units hired and sold across MonEquip branches in Victoria. From the visual, it's clear that the Clayton branch recorded the highest level of activity, while Cheltenham showed the least. Generally, most branches reported more equipment hires than sales, with the exception of Clayton (1,075 sold vs. 1,025 hired) and Parkville (493 sold vs. 461 hired). However, the difference in these figures

remains minimal. Overall, the data indicates that hiring transactions slightly outpace sales transactions across MonEquip's branches.

2. *Revenue generation through the years (2018-2020)*

Code:

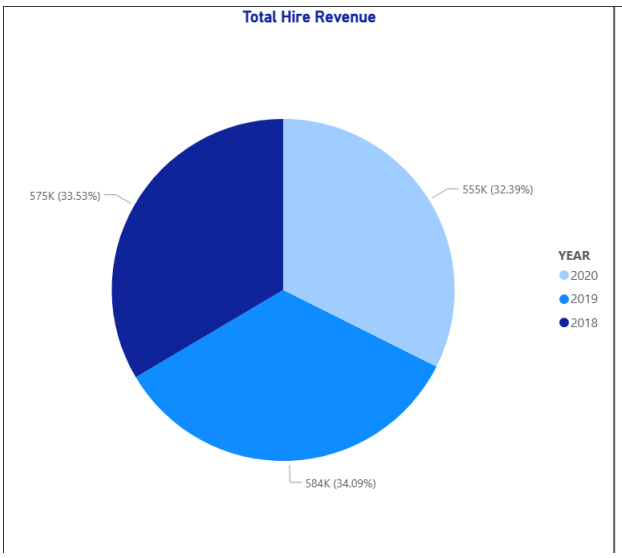
```
-- OLAP Operation : Selling Revenue  by year
SELECT
    t.year,
    SUM(sf.total_sales_revenue) AS total_sales_revenue,
    ROUND(
        SUM(sf.total_sales_revenue) * 100 /
        SUM(SUM(sf.total_sales_revenue)) OVER(),
        2
    ) AS percentage_contribution
FROM SalesFact sf
JOIN TimeDIM t
    ON sf.time_id = t.time_id
GROUP BY t.year
ORDER BY t.year;

-- OLAP Operation : Hiring Revenue by year
SELECT
    t.year,
    SUM(hf.total_hire_revenue) AS total_hire_revenue,
    ROUND(
        SUM(hf.total_hire_revenue) * 100 /
        SUM(SUM(hf.total_hire_revenue)) OVER(),
        2
    ) AS percentage_contribution
FROM HireFACT hf
JOIN TimeDIM t
    ON hf.time_id = t.time_id
GROUP BY t.year
ORDER BY t.year;
```

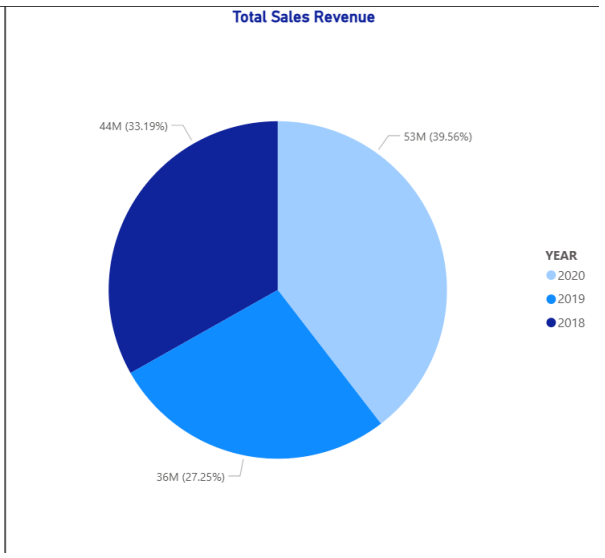
Output:

	YEAR	TOTAL_SALES_REVENUE	PERCENTAGE_CONTRIBUTION
1	2018	44313200	33.19
2	2019	36384050	27.25
3	2020	52817800	39.56

	YEAR	TOTAL_HIRE_REVENUE	PERCENTAGE_CONTRIBUTION
1	2018	574850	33.53
2	2019	584495	34.09
3	2020	555335	32.39

Visualisation:





From the pie charts above, we can observe how hire and sales revenue have changed over the three years from 2018 to 2020. In 2018, there was little difference between the revenue generated from hired and sold equipment. By 2019, sales revenue had dipped below hire revenue, but the most notable shift occurred in 2020, when sales revenue surpassed hire revenue by a significant margin. Overall, hire revenue remained relatively stable across the three years, while sales revenue in 2020 showed a sharp increase compared to the previous years.

3. *Product Category Analysis*

Code:

```
SELECT
    c.category_description,
    SUM(hf.total_hire_revenue) AS total_hire_revenue,
    ROUND(
        SUM(hf.total_hire_revenue) * 100 /
        SUM(SUM(hf.total_hire_revenue)) OVER(),
        2
    ) AS percentage_contribution
FROM HireFact hf
JOIN CategoryDIM c
    ON hf.category_id = c.category_id
GROUP BY c.category_description
ORDER BY total_hire_revenue DESC;

-- OLAP Operation : Sales Revenue by Category
SELECT
```

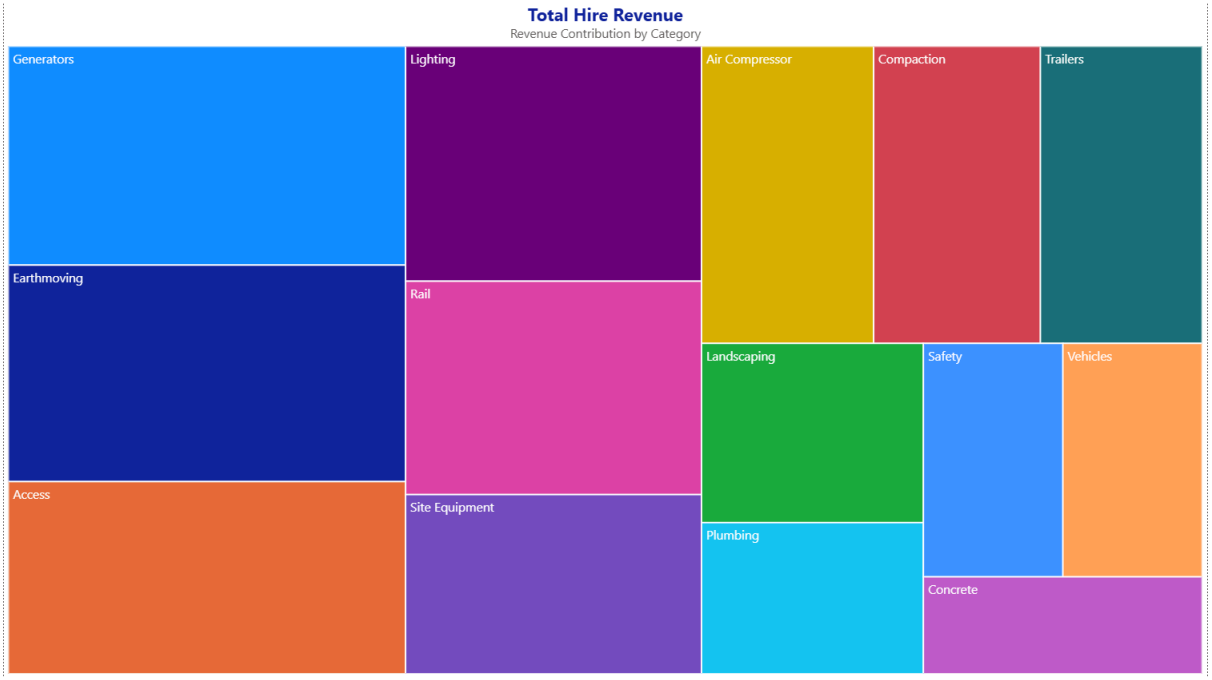
```
    c.category_description,
    SUM(sf.total_sales_revenue) AS total_sales_revenue,
    ROUND(
        SUM(sf.total_sales_revenue) * 100 /
        SUM(SUM(sf.total_sales_revenue)) OVER(),
        2
    ) AS percentage_contribution
FROM SalesFact sf
JOIN CategoryDIM c
    ON sf.category_id = c.category_id
GROUP BY c.category_description
ORDER BY total_sales_revenue DESC;
```

Output:

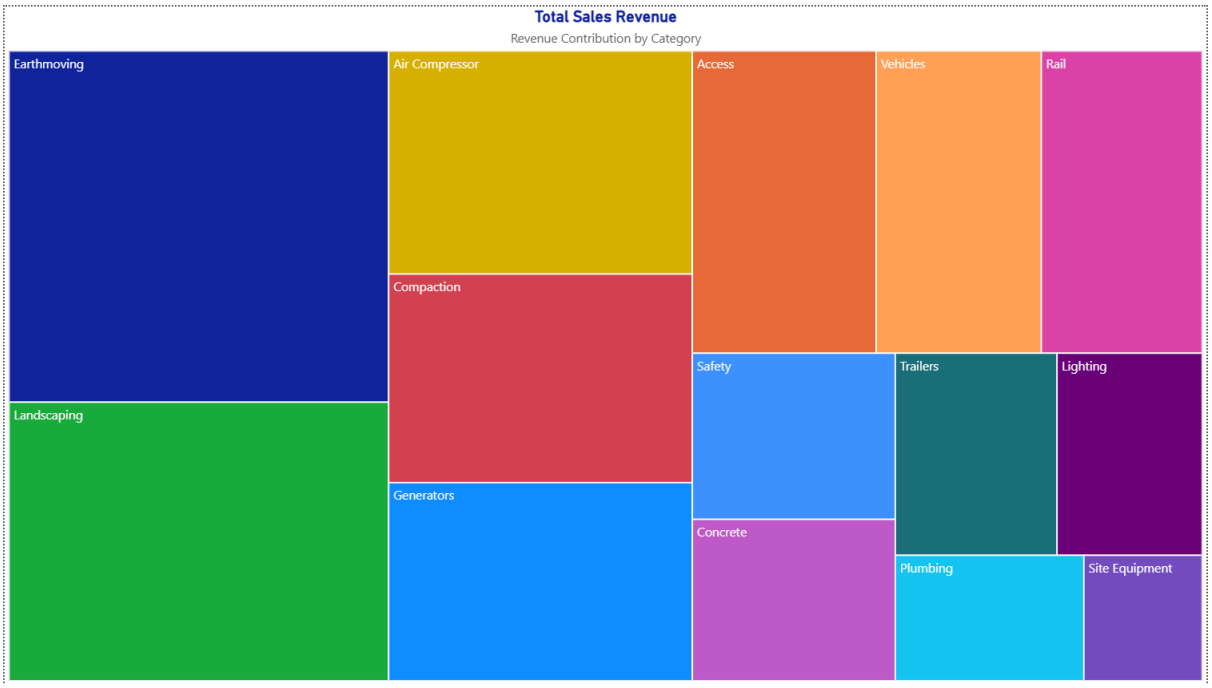
	CATEGORY_DESCRIPTION	TOTAL_HIRE_REVENUE	PERCENTAGE_CONTRIBUTION
1	Generators	199040	11.61
2	Earthmoving	196930	11.48
3	Access	174710	10.19
4	Lighting	159140	9.28
5	Rail	144580	8.43
6	Site Equipment	121345	7.08
7	Air Compressor	117065	6.83
8	Compaction	113370	6.61
9	Trailers	109980	6.41
10	Landscaping	91040	5.31
11	Plumbing	76720	4.47
12	Safety	74560	4.35
13	Vehicles	74410	4.34
14	Concrete	61790	3.6

	CATEGORY_DESCRIPTION	TOTAL_SALES_REVENUE	PERCENTAGE_CONTRIBUTION
1	Earthmoving	23690000	17.74
2	Landscaping	18789200	14.07
3	Air Compressor	12037000	9.02
4	Compaction	11259200	8.43
5	Generators	10690500	8.01
6	Access	9872250	7.39
7	Vehicles	8872000	6.64
8	Rail	8632300	6.47
9	Safety	5995000	4.49
10	Concrete	5820600	4.36
11	Trailers	5808800	4.35
12	Lighting	5211000	3.9
13	Plumbing	4202600	3.15
14	Site Equipment	2634600	1.97

Visualisation:



The top three product categories contributing to hire revenue over the years were Generators, Earthmoving, and Access, while Concrete consistently contributed the least.



On the sales side, the two major contributors to revenue were Earthmoving and Landscaping, whereas Site Equipment generated comparatively lower sales revenue.

Overall, Generators and Access show relatively lower contributions on the sales front, whereas Concrete demonstrates a stronger performance in sales compared to hire revenue.