MONASH University

FIT3003 Business Intelligence & Data Warehousing

Assessment 2: Major Assignment: BI & DW (40%)

Semester 2, 2025

Due: Friday, 17 October 2025, 8.55 PM

Zoe Yow Cui Yi 33214476

Table of Contents

A. Transformation Stage 4
Preparation Stage 4
Exploring the Operational Database 4
Data Cleaning 5
Error 1: Duplicated records in CUSTOMER table 5
Error 2: Duplicated records in CUSTOMER_TYPE table 6
Error 3: Null value in CATEGORY_DESCRIPTION column of CATEGORY table 7
Error 4: Inconsistent values in GENDER column of CUSTOMER Table 8
Error 5: Invalid EQUIPMENT_ID and CUSTOMER_ID values in HIRE table 9
Error 6: Incorrect TOTAL_HIRE_PRICE value in HIRE table 10
Error 7: Inconsistent START_DATE and END_DATE values in HIRE table 11
Error 8: Invalid STAFF_ID value in HIRE table 11
Error 9: Invalid CATEGORY_ID value in EQUIPMENT table (due to Error 3) 13
Error 10: Incorrect QUANTITY value in SALES table 14
Design Task A 15
Design Task B 15
Explanation of the Differences among SCD Types 1, 2, 3, 4, and 6 17
Star/Snowflake Implementation for Design Task A 18
Table Structure of Dimension Tables 18
Customer Type Dimension Table 18
Time Dimension Table 18
Season Dimension Table 18
Sales Price Scale Dimension Table 18
Category Dimension Table 19
Company Branch Dimension Table 19
Table Structure of Fact Tables 19
Sales Fact Table 19
Hire Fact Table 19
B. Data Analytic Stage 19
Findings Report 19
Overall Revenue Performance 19

Number of Sales by Sales Price Scale (Low/Medium/High) 21
Total Sales and Hire Revenue by Season 22
Top 5 Company Branches with the Highest Total Sales Revenue 23
Average Sales/Hire Revenue per Sale/Hire Transaction by Customer Type 23
Top 5 Most Hired Equipment Category 24
Appendix 25
SQL Statements for Dimension Tables Implementation 25
SQL Statements for Fact Tables Implementation 27

A. Transformation Stage

Preparation Stage

Exploring the Operational Database

Firstly, make a copy of each table from the operational database

```
-- Drop tables
DROP TABLE ADDRESS CASCADE CONSTRAINT PURGE;
DROP TABLE CATEGORY CASCADE CONSTRAINT PURGE;
DROP TABLE CUSTOMER CASCADE CONSTRAINT PURGE;
DROP TABLE CUSTOMER_TYPE CASCADE CONSTRAINT PURGE;
DROP TABLE EQUIPMENT CASCADE CONSTRAINT PURGE;
DROP TABLE HIRE CASCADE CONSTRAINT PURGE;
DROP TABLE SALES CASCADE CONSTRAINT PURGE;
DROP TABLE STAFF CASCADE CONSTRAINT PURGE;
DROP TABLE CUSTOMER_CLEAN CASCADE CONSTRAINT PURGE;

-- Create tables
CREATE TABLE ADDRESS
AS SELECT * FROM MONEQUIP.ADDRESS;

CREATE TABLE CATEGORY
AS SELECT * FROM MONEQUIP.CATEGORY;

CREATE TABLE CUSTOMER
AS SELECT * FROM MONEQUIP.CUSTOMER;

CREATE TABLE CUSTOMER_TYPE
AS SELECT * FROM MONEQUIP.CUSTOMER_TYPE;

CREATE TABLE EQUIPMENT
AS SELECT * FROM MONEQUIP.EQUIPMENT;

CREATE TABLE HIRE
AS SELECT * FROM MONEQUIP.HIRE;

CREATE TABLE SALES
AS SELECT * FROM MONEQUIP.SALES;

CREATE TABLE STAFF
AS SELECT * FROM MONEQUIP.STAFF;
```

Next, explore the operational database through the tables created using "`SELECT *`" and "`SELECT COUNT(*)`" statements.

```
SELECT * FROM ADDRESS;
SELECT * FROM CATEGORY;
SELECT * FROM CUSTOMER;
SELECT * FROM CUSTOMER_TYPE;
SELECT * FROM EQUIPMENT;
SELECT * FROM HIRE;
SELECT * FROM SALES;
SELECT * FROM STAFF;

SELECT COUNT(*) FROM ADDRESS;
SELECT COUNT(*) FROM CATEGORY;
SELECT COUNT(*) FROM CUSTOMER;
SELECT COUNT(*) FROM CUSTOMER_TYPE;
SELECT COUNT(*) FROM EQUIPMENT;
SELECT COUNT(*) FROM HIRE;
SELECT COUNT(*) FROM SALES;
SELECT COUNT(*) FROM STAFF;
```

Data Cleaning

The strategies used in the process includes checking data duplication between records, relationship problems, inconsistent values at a record, incorrect value problem, and null value problems.

Error 1: Duplicated records in CUSTOMER table

This error was found using the following SQL statement:

```
SELECT CUSTOMER_ID, COUNT(*)
FROM CUSTOMER
GROUP BY CUSTOMER_ID
HAVING COUNT(*) > 1;
```

Here is the output of the SQL statement:

Before cleaning, this table's content and number of records were checked using the following SQL statements:

```
SELECT * FROM CUSTOMER;
SELECT COUNT(*) FROM CUSTOMER;
```

Here is the output of these SQL statement:

All rows fetched: 153 in 0.826 seconds

	CUSTOMER_ID	CUSTOMER_TYPE_ID	NAME	GENDER	ADDRESS_ID	PHONE	EMAIL
146	146	2	Shayla Curtiss	Female	146	772 408 7008	scurtiss41@squidoo.com
147	147	2	Ilsa Cawthron	Female	147	475 279 1818	icawthron42@wikispaces.com
148	148	1	Katina Scotter	Female	148	384 698 7868	kscotter43@tmall.com
149	149	2	Britte Nickols	Female	149	955 251 7916	bnickols44@sina.com.cn
150	150	2	Martainn Sidon	Male	150	780 490 3276	msidon45@imageshack.us
151	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu
152	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu
153	52	2	Abbie Maddie	Male	52	904 627 9038	amaddie1f@columbia.edu

All rows fetched: 1 in 0.191 seconds

	COUNT(*)
1	153

To clean this table, the following SQL statement was used:

```
CREATE TABLE CUSTOMER_CLEAN AS
SELECT DISTINCT *
FROM CUSTOMER;
```

The same SQL statements that were used before cleaning were used after cleaning to check the table's content and number of records.

After cleaning, here is the output of the SQL statements:

All rows fetched: 150 in 0.824 seconds

	CUSTOMER_ID	CUSTOMER_TYPE_ID	NAME	GENDER	ADDRESS_ID	PHONE	EMAIL
143	87	2	Wilma Abyss	F	87	888 141 6586	wabyss2e@xinhuanet.com
144	103	1	Pearline Haacker	Female	103	985 106 8056	phaacker2u@virginia.edu
145	112	2	Lek Darte	Male	112	789 267 8267	ldarte33@jugem.jp
146	118	2	Holt Thonason	Male	118	431 335 1570	hthonason39@moonfruit.com
147	124	2	Lea Bartoletti	Female	124	817 793 4925	lbartoletti3f@dyndns.org
148	126	2	Armstrong Bown	Male	126	758 633 4898	abown3h@bing.com
149	136	1	Cory Ruckledge	Male	136	690 832 6051	cruckledge3r@dropbox.com
150	140	2	Junette Bartul	Female	140	429 869 8287	jbartul3v@unc.edu

All rows fetched: 1 in 0.231 seconds

	COUNT(*)
1	150

Error 2: Duplicated records in CUSTOMER_TYPE table

This error was found using the following SQL statement:

```
SELECT CUSTOMER_TYPE_ID, COUNT(*)
FROM CUSTOMER_TYPE
GROUP BY CUSTOMER_TYPE_ID
HAVING COUNT(*) > 1;
```

Here is the output of the SQL statement:

All rows fetched: 1 in 0.214 seconds

	CUSTOMER_TYPE_ID	COUNT(*)
1	2	2

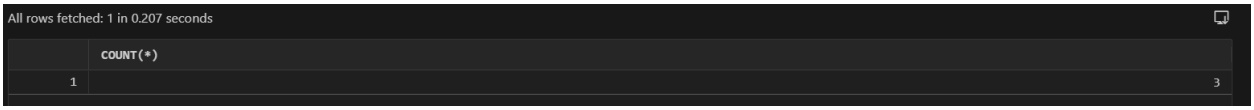
Before cleaning this table, the table's content and the number of records were checked using the following SQL statement:

```
SELECT * FROM CUSTOMER_TYPE;
SELECT COUNT(*) FROM CUSTOMER_TYPE;
```

Here is the output of the SQL statements:

All rows fetched: 3 in 0.166 seconds

	CUSTOMER_TYPE_ID	DESCRIPTION
1	1	Individual
2	2	Business
3	2	business



All rows fetched: 1 in 0.207 seconds

	COUNT(*)
1	3

To clean this table, the following SQL statement was used:

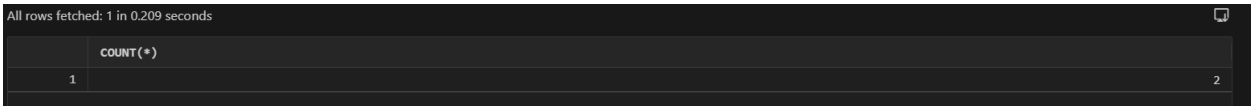
```
DELETE FROM CUSTOMER_TYPE
WHERE CUSTOMER_TYPE_ID = 2
    AND DESCRIPTION = 'business';
```

The same SQL statements that were used before cleaning were used after cleaning to check the table's content and number of records.

After cleaning this table, here is the output of the SQL statement:

All rows fetched: 2 in 0.184 seconds

	CUSTOMER_TYPE_ID	DESCRIPTION
1	1	Individual
2	2	Business



All rows fetched: 1 in 0.209 seconds

	COUNT(*)
1	2

Error 3: Null value in CATEGORY_DESCRIPTION column of CATEGORY table

This error was found using the following SQL statement:

```
SELECT * FROM CATEGORY
WHERE CATEGORY_DESCRIPTION = 'null';
```

Here is the output of the SQL statement:

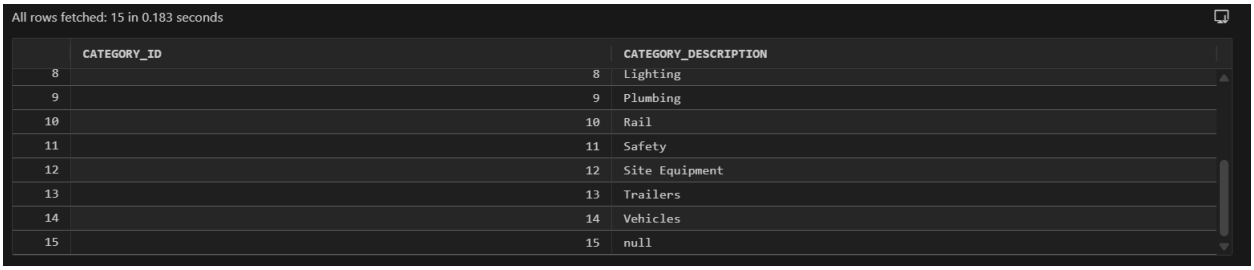
All rows fetched: 1 in 0.131 seconds

	CATEGORY_ID	CATEGORY_DESCRIPTION
1	15	null

Before cleaning this table, its contents and the number of records were checked using the following SQL statements:

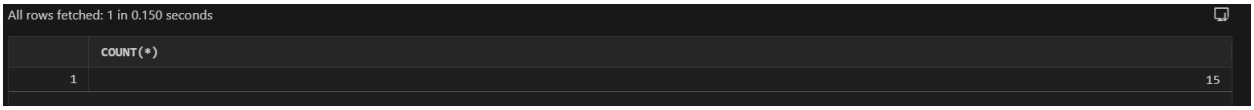
```
SELECT * FROM CATEGORY;
SELECT COUNT(*) FROM CATEGORY;
```

Here is the output of the SQL statements:



All rows fetched: 15 in 0.183 seconds

	CATEGORY_ID	CATEGORY_DESCRIPTION
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



All rows fetched: 1 in 0.150 seconds

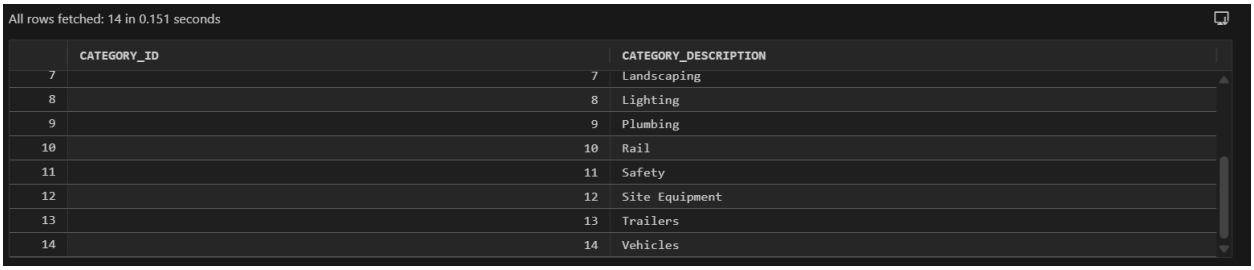
	COUNT(*)
1	15

To clean this table, the following SQL statement was used:

```
DELETE FROM CATEGORY
WHERE CATEGORY_DESCRIPTION = 'null';
```

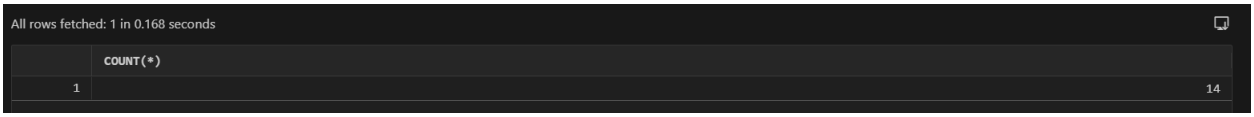
After cleaning this table, its content and number of records using the same SQL statements used when checking the table before cleaning it.

Here is the output:



All rows fetched: 14 in 0.151 seconds

	CATEGORY_ID	CATEGORY_DESCRIPTION
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



All rows fetched: 1 in 0.168 seconds

	COUNT(*)
1	14

Error 4: Inconsistent values in GENDER column of CUSTOMER Table

This error was found using the following SQL statement:

```
SELECT DISTINCT GENDER
FROM CUSTOMER;
```

Here is the output of the SQL statement:

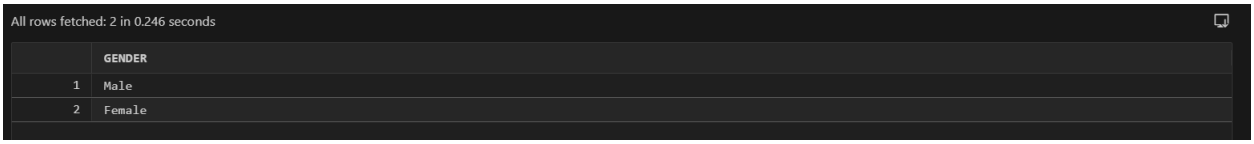
All rows fetched: 4 in 0.172 seconds

	GENDER
1	Male
2	M
3	Female
4	F

To clean this table, the following SQL statement was used:

```
UPDATE CUSTOMER
SET GENDER = CASE
              WHEN UPPER(GENDER) = 'M' THEN 'Male'
              WHEN UPPER(GENDER) = 'F' THEN 'Female'
             END
WHERE UPPER(GENDER) IN ('M', 'F');
```

After cleaning this table, here is the output from the same SQL statement used to find this error:



All rows fetched: 2 in 0.246 seconds

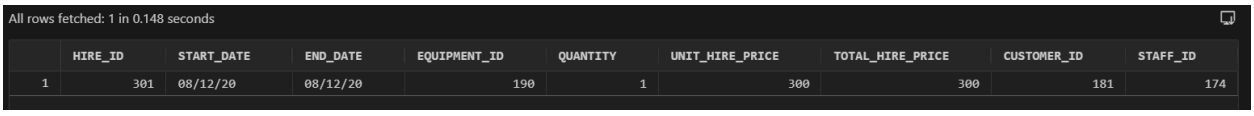
	GENDER
1	Male
2	Female

Error 5: Invalid EQUIPMENT_ID and CUSTOMER_ID values in HIRE table

The error was found using the following SQL statements:

```
SELECT *
FROM HIRE
WHERE EQUIPMENT_ID NOT IN
    (SELECT EQUIPMENT_ID
    FROM EQUIPMENT);

SELECT *
FROM HIRE
WHERE CUSTOMER_ID NOT IN
    (SELECT CUSTOMER_ID
    FROM CUSTOMER_CLEAN);
```

Here is the output of these SQL statements:



All rows fetched: 1 in 0.148 seconds

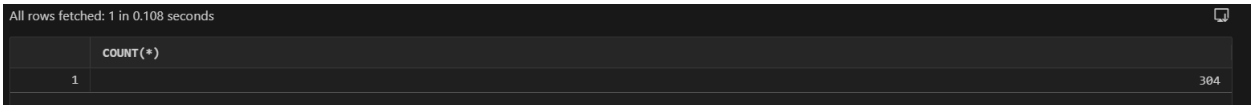
	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID
1	301	08/12/20	08/12/20	190	1	300	300	181	174

Both outputs are found in the same record.

Before cleaning the number of records were checked using the following SQL statement:

```
SELECT COUNT(*) FROM HIRE;
```

Here is the output of the SQL statement:



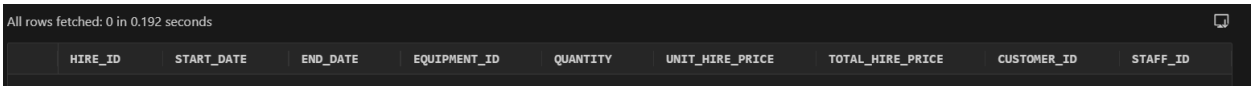
To clean this table from this error, the following SQL statement was used:

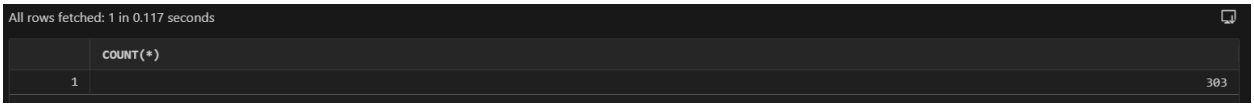
```
UPDATE HIRE
SET EQUIPMENT_ID = NULL, CUSTOMER_ID = NULL
WHERE EQUIPMENT_ID NOT IN
    (SELECT EQUIPMENT_ID
    FROM EQUIPMENT)
    OR CUSTOMER_ID NOT IN
    (SELECT CUSTOMER_ID
    FROM CUSTOMER_CLEAN);

DELETE FROM HIRE
WHERE EQUIPMENT_ID IS NULL
    OR CUSTOMER_ID IS NULL;
```

After cleaning the error from this table, the following SQL statements were used to check the table:

```
SELECT *
FROM HIRE
WHERE EQUIPMENT_ID NOT IN
    (SELECT EQUIPMENT_ID
    FROM EQUIPMENT)
    OR CUSTOMER_ID NOT IN
    (SELECT CUSTOMER_ID
    FROM CUSTOMER_CLEAN);

SELECT COUNT(*) FROM HIRE;
```

Here is the output:





Error 6: Incorrect TOTAL_HIRE_PRICE value in HIRE table

This error was found using the following SQL statement:

```
SELECT *
FROM HIRE
WHERE TOTAL_HIRE_PRICE < 0;
```

Here is the output of the SQL statement:

All rows fetched: 2 in 0.110 seconds

	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID
1	303	25/01/90	27/12/99	43	3	50	-150	53	223
2	304	08/12/20	08/12/20	114	1	350	-1	34	85

Before cleaning this error from this table, the number of records was checked using the following SQL statement:

```
SELECT COUNT(*) FROM HIRE;
```

Here is the output of the SQL statement:

All rows fetched: 1 in 0.119 seconds

	COUNT(*)
1	303

Please refer to Error 8 on the process of cleaning this error.

Error 7: Inconsistent START_DATE and END_DATE values in HIRE table

This error was found using the following SQL statement:

```
SELECT *
FROM HIRE
WHERE START_DATE < TO_DATE('1-APR-2018')
    OR END_DATE > TO_DATE('31-DEC-2020')
    or START_DATE > END_DATE;
```

Here is the output of the SQL statement:

All rows fetched: 2 in 0.110 seconds

	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID
1	303	25/01/90	27/12/99	43	3	50	-150	53	223
2	302	05/12/20	17/10/20	21	2	100	200	111	123

Before cleaning this error from this table, the number of records were checked using the following SQL statement:

```
SELECT COUNT(*) FROM HIRE;
```

Here is the output of the SQL statement:

All rows fetched: 1 in 0.119 seconds

	COUNT(*)
1	303

Please refer to Error 8 on the process of cleaning this error.

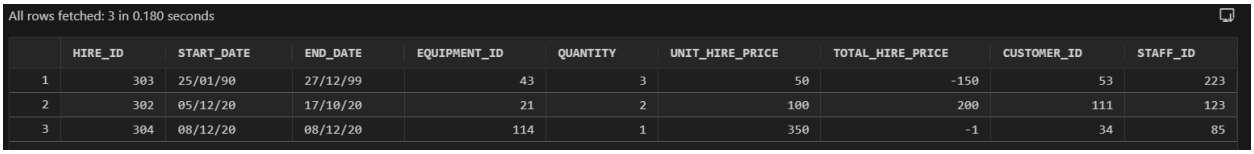
Error 8: Invalid STAFF_ID value in HIRE table

This error was found using the following SQL statement:

```
SELECT *
```

```
FROM HIRE
WHERE STAFF_ID NOT IN
    (SELECT STAFF_ID
    FROM STAFF);
```

Here is the output of the SQL statement:



All rows fetched: 3 in 0.180 seconds

	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID
1	303	25/01/90	27/12/99	43	3	50	-150	53	223
2	302	05/12/20	17/10/20	21	2	100	200	111	123
3	304	08/12/20	08/12/20	114	1	350	-1	34	85

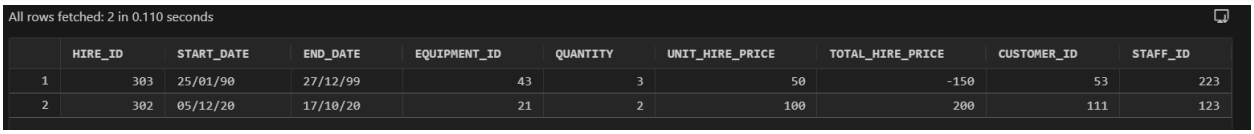
Before cleaning this table, the number of records were checked using the following SQL statement:

```
SELECT COUNT(*) FROM HIRE;
```

Here is the output of the SQL statement:

All rows fetched: 1 in 0.119 seconds

	COUNT(*)
1	303



All rows fetched: 2 in 0.110 seconds

	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID
1	303	25/01/90	27/12/99	43	3	50	-150	53	223
2	302	05/12/20	17/10/20	21	2	100	200	111	123

This error contains the same records that error 6 and 7 points to, where error 6 points to HIRE_ID values 303 and 304, and error 7 points to HIRE_ID values 302 and 303.

Therefore, to clean this table from error 6, 7, and 8, the following SQL statements were used:

```
UPDATE HIRE
SET STAFF_ID = NULL
WHERE STAFF_ID NOT IN
    (SELECT STAFF_ID
    FROM STAFF);

DELETE FROM HIRE
WHERE STAFF_ID IS NULL;
```

After cleaning this table, the following SQL statements were used to check the table:

```
SELECT *
FROM HIRE
WHERE STAFF_ID NOT IN
    (SELECT STAFF_ID
    FROM STAFF);

SELECT COUNT(*) FROM HIRE;
```

Here is the output:

All rows fetched: 0 in 0.749 seconds

	HIRE_ID	START_DATE	END_DATE	EQUIPMENT_ID	QUANTITY	UNIT_HIRE_PRICE	TOTAL_HIRE_PRICE	CUSTOMER_ID	STAFF_ID

All rows fetched: 1 in 0.116 seconds

	COUNT(*)
1	300

Error 9: Invalid CATEGORY_ID value in EQUIPMENT table (due to Error 3)

This error was found using the following SQL statement:

```
SELECT *
FROM EQUIPMENT
WHERE CATEGORY_ID NOT IN
    (SELECT CATEGORY_ID
    FROM CATEGORY);
```

Here is the output of the SQL statement:

All rows fetched: 1 in 0.293 seconds

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

Before cleaning this table, the number of records in this table was checked using the following SQL statement:

```
SELECT COUNT(*) FROM EQUIPMENT;
```

Here is the output of the SQL statement:

All rows fetched: 1 in 0.274 seconds

	COUNT(*)
1	158

To clean this table, the following SQL statements were used:

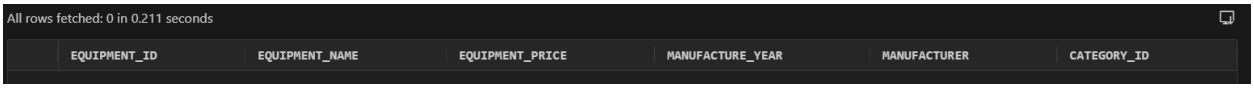
```
UPDATE EQUIPMENT
SET CATEGORY_ID = NULL
WHERE CATEGORY_ID NOT IN
    (SELECT CATEGORY_ID
    FROM CATEGORY);

DELETE FROM EQUIPMENT
WHERE CATEGORY_ID IS NULL;
```

After cleaning this table, the following SQL statements were used to check the table:

```
SELECT *
FROM EQUIPMENT
WHERE CATEGORY_ID NOT IN
    (SELECT CATEGORY_ID
```

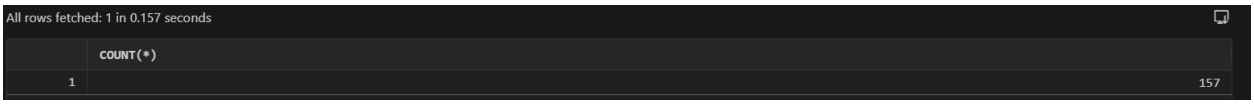
```
    FROM CATEGORY);

SELECT COUNT(*) FROM EQUIPMENT;
```

Here is the output:



All rows fetched: 0 in 0.211 seconds

EQUIPMENT_ID	EQUIPMENT_NAME	EQUIPMENT_PRICE	MANUFACTURE_YEAR	MANUFACTURER	CATEGORY_ID



All rows fetched: 1 in 0.157 seconds

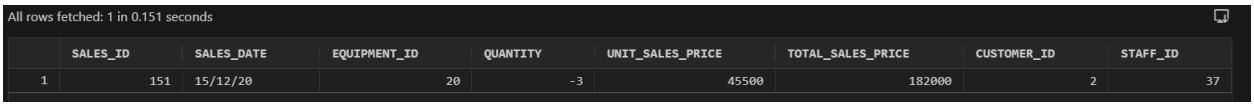
	COUNT(*)
1	157

Error 10: Incorrect QUANTITY value in SALES table

This error was found using the following SQL statement:

```
SELECT *
FROM SALES
WHERE QUANTITY < 0;
```

Here is the output of the SQL statement:



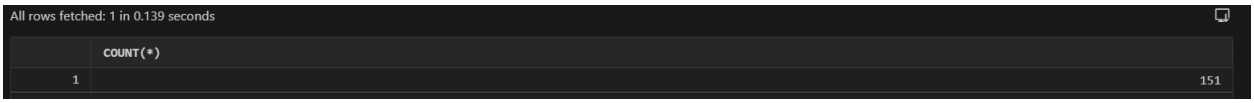
All rows fetched: 1 in 0.151 seconds

	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

Before cleaning this table, the number of records in the table was checked using the following SQL statement:

```
SELECT COUNT(*) FROM SALES;
```

Here is the output of the SQL statement:



All rows fetched: 1 in 0.139 seconds

	COUNT(*)
1	151

To clean this table, the following SQL statement was used:

```
UPDATE SALES
SET QUANTITY = NULL
WHERE QUANTITY < 0;

DELETE FROM SALES
WHERE QUANTITY IS NULL;
```

After cleaning this table, the following SQL statements were used to check the table:

```
SELECT *
FROM SALES
WHERE QUANTITY < 0;
```

```
SELECT COUNT(*) FROM SALES;
```

Here is the output:

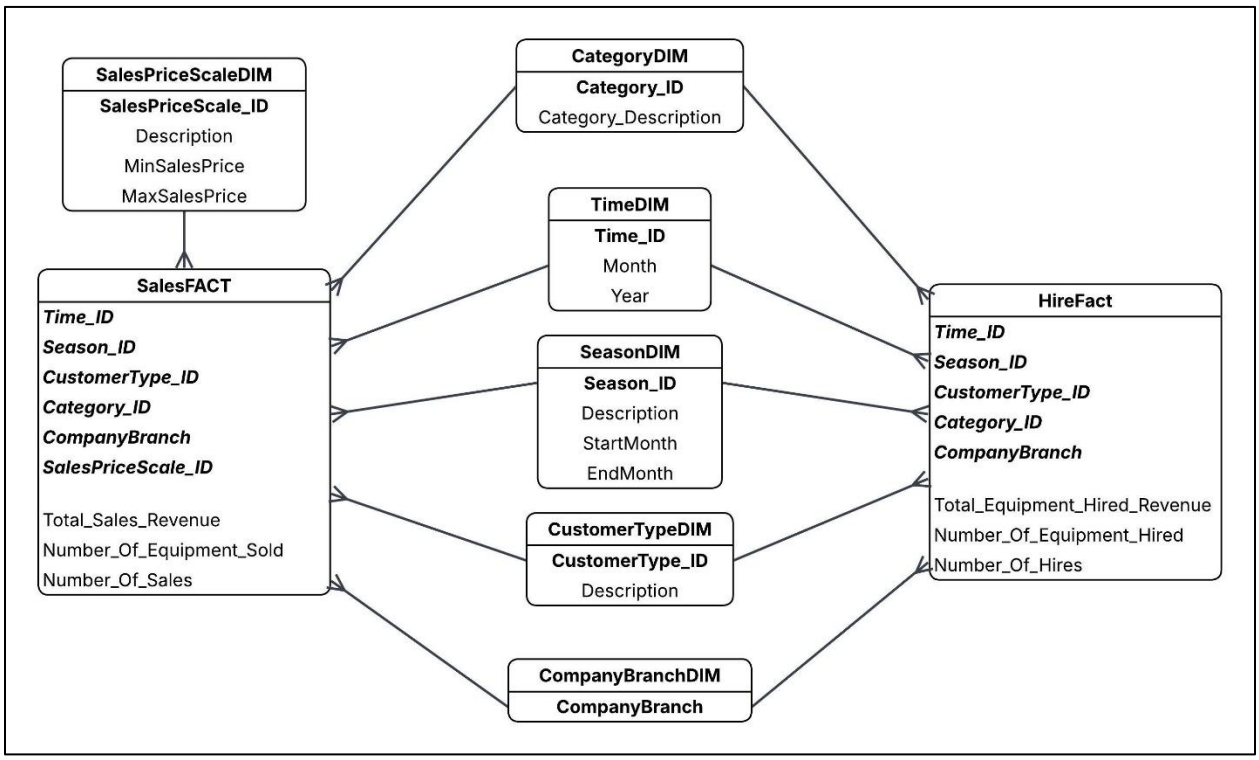
All rows fetched: 0 in 0.183 seconds

	SALES_ID	SALES_DATE	EQUIPMENT_ID	QUANTITY	UNIT_SALES_PRICE	TOTAL_SALES_PRICE	CUSTOMER_ID	STAFF_ID

All rows fetched: 1 in 0.147 seconds

	COUNT(*)
1	150

Design Task A

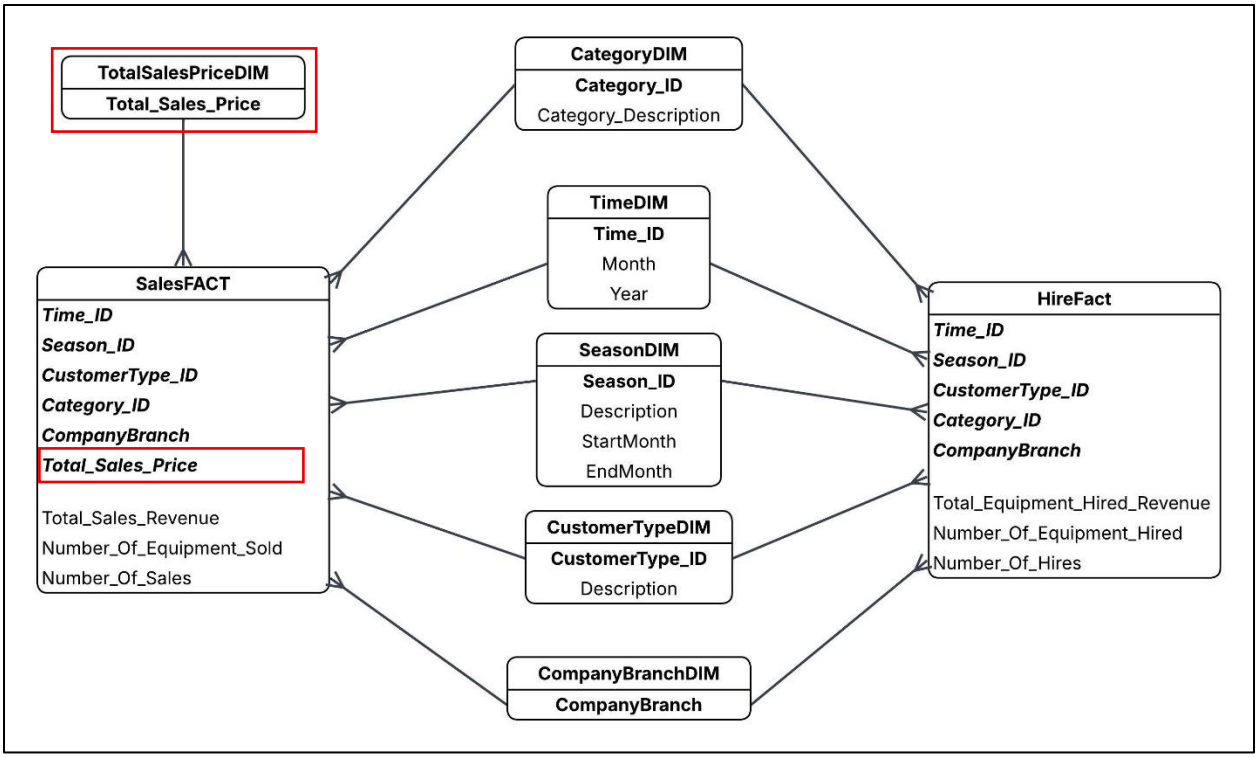


Design Task B

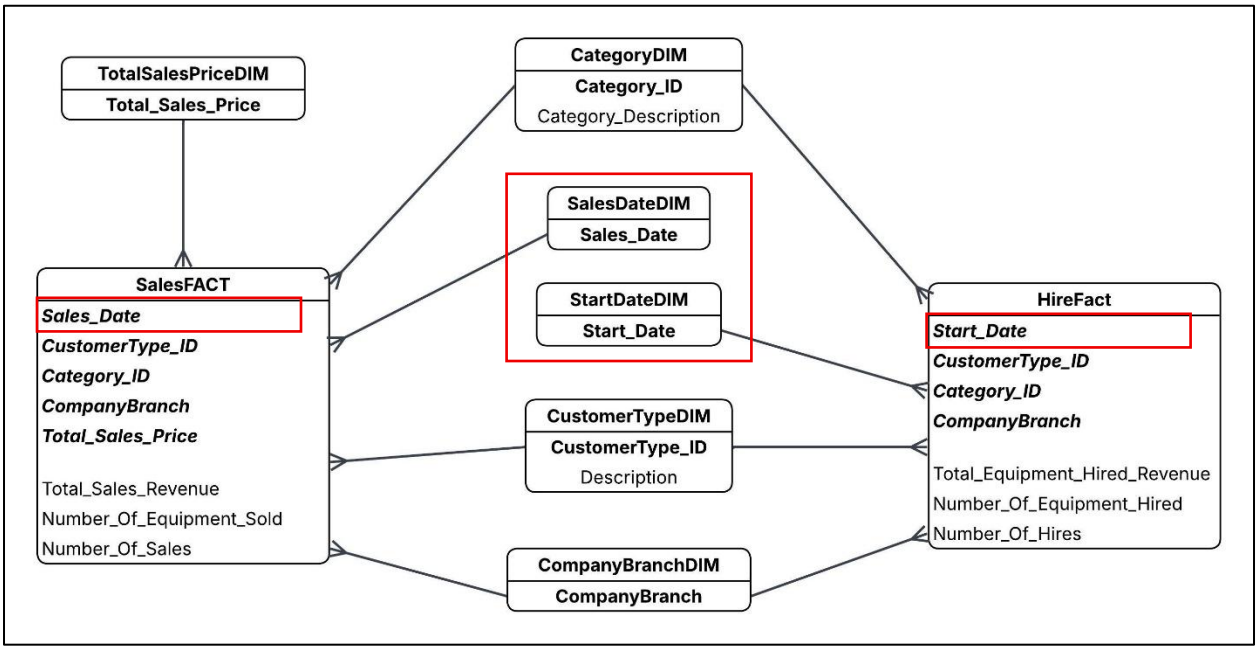
There are 3 aggregated dimensions: SalesPriceScaleDIM, TimeDIM, and SeasonDIM.

There are several ways to increase the granularity of Design Task A's fact tables.

1. SalesPriceScaleDIM can be changed to TotalSalesPriceDIM
   Here is an example star schema of this change:



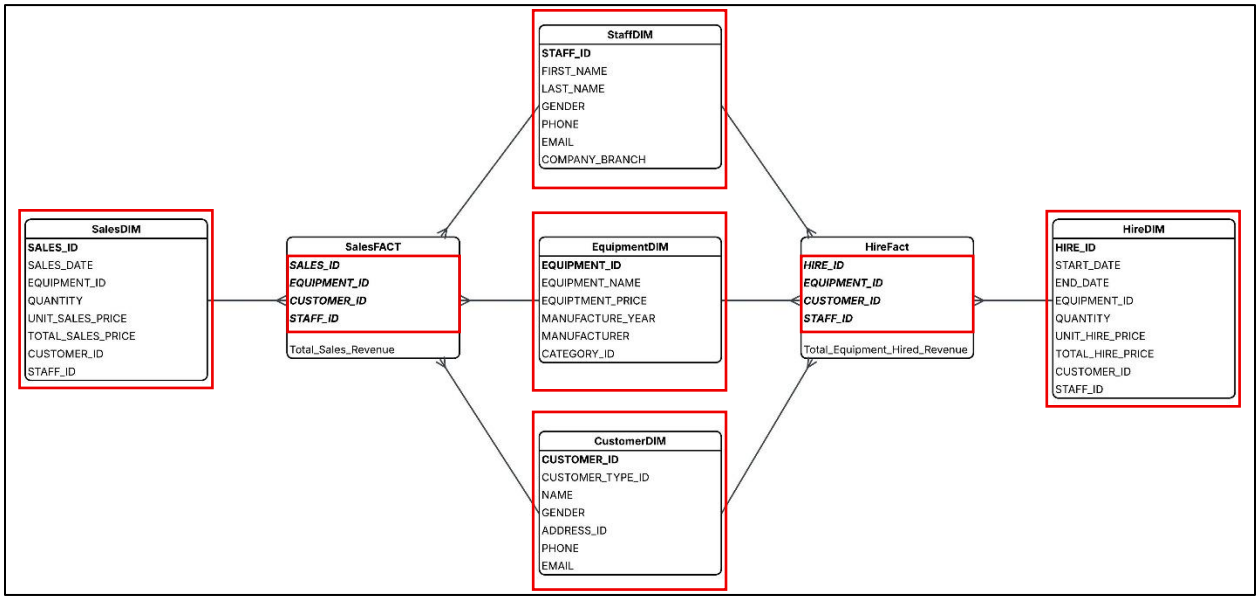
2. TimeDIM and SeasonDIM can be changed to two dimensions including StartDateDIM and SalesDateDIM
   Here is an example star schema for this change:



3. All three aggregated dimensions can be replaced by SalesDIM and HireDIM.
4. CompanyBranchDIM can be replaced by StaffDIM
5. CustomerTypeDIM can be replaced by CustomerDIM

6. CategoryDIM can be replaced by EquipmentDIM

Here is the example star schema for changes 3, 4, 5, and 6:



Explanation of the Differences among SCD Types 1, 2, 3, 4, and 6

Slowly Changing Dimensions or SCD are dimensions where the records of these dimensions change slowly over a period of time. SCDs serve to preserve historical data and monitor how a dimension's attributes change over time. There are a few types of SCDs which includes Type 1, 2, 3, 4, and 6.

Type 1 records the latest value of records but does not record any history of changes in a dimension. Therefore, the present state is always reflected in the dimension.

However, Type 2 keeps track of the history. The history remains in the main dimension with the dates. The dimension table is updated whenever there is a change, and a new row is inserted to represent the new data.

Besides that, type 3 is a simplified version of type 2 and a mix of type 1 and type 2. Type 3 keeps historical data but unlike type 2 which maintains the entire history of a record’s changes, it only keeps the last two changes in two separate columns, i.e., “previous value” and “current value” columns.

Moreover, type 4 preserves historical data by generating independent tables that record changes to particular attributes. The primary dimension table is still mostly the same, however linked tables are where the changes are recorded.

Lastly, SCD type 6 is a combination of SCD type 2 and 3. Where SCD type 3 does not record all the history and SCD type 2 requires a separate identifier, SCD type 6 does not require a separate

identifier and records all the history. Additionally, no separate history table needs to be kept for SCD Type 6. The original dimension table retains the history itself.

Giving an example, for Design Task A, the temporal dimension can be represented by the TimeDIM table, which includes attributes such as Time_ID, Month, and Year. This dimension is primarily used for aggregating data by time periods, such as months or years. Since time-related values generally remain static, SCD Type 1 is most appropriate because overwriting is sufficient. However, if fiscal calendars or seasonal mappings change over time, SCD Type 2 may be considered to preserve historical accuracy.

Nevertheless, no SCD types were implemented because none of the attributes within the dimensions are expected to change over time. The data being managed is constant, meaning historical tracking is unnecessary. Therefore, applying SCD techniques would not provide any additional value in this context.

Star/Snowflake Implementation for Design Task A

Table Structure of Dimension Tables

The SQL statements used to create the dimension tables can be found in the Appendix.

Customer Type Dimension Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CUSTOMER_TYPE_ID	NUMBER	Yes	(null)	1	(null)
2	DESCRIPTION	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)

Time Dimension Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TIME_ID	VARCHAR2(6 BYTE)	Yes	(null)	1	(null)
2	TIME_MONTH	VARCHAR2(2 BYTE)	Yes	(null)	2	(null)
3	TIME_YEAR	VARCHAR2(4 BYTE)	Yes	(null)	3	(null)

Season Dimension Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SEASON_ID	NUMBER	Yes	(null)	1	(null)
2	SEASON_DESC	VARCHAR2(10 BYTE)	Yes	(null)	2	(null)
3	START_MONTH	NUMBER	Yes	(null)	3	(null)
4	END_MONTH	NUMBER	Yes	(null)	4	(null)

Sales Price Scale Dimension Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SALESPRICESCALE_ID	NUMBER	Yes	(null)	1	(null)
2	SALESPRICESCALE_DESC	VARCHAR2(10 BYTE)	Yes	(null)	2	(null)
3	MINSALESPRICE	NUMBER	Yes	(null)	3	(null)
4	MAXSALESPRICE	NUMBER	Yes	(null)	4	(null)

Category Dimension Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CATEGORY_ID	NUMBER	Yes	(null)	1	(null)
2	CATEGORY_DESCRIPTION	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)

Company Branch Dimension Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COMPANY_BRANCH	VARCHAR2(20 BYTE)	Yes	(null)	1	(null)

Table Structure of Fact Tables

The SQL statements used to create the fact tables can be found in the Appendix.

Sales Fact Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TIME_ID	NUMBER	Yes	(null)	1	(null)
2	SEASON_ID	NUMBER	Yes	(null)	2	(null)
3	CUSTOMER_TYPE_ID	NUMBER	Yes	(null)	3	(null)
4	CATEGORY_ID	NUMBER	Yes	(null)	4	(null)
5	COMPANY_BRANCH	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)
6	SALESPRICESCALE_ID	NUMBER	Yes	(null)	6	(null)
7	TOTAL_SALES_REVENUE	NUMBER	Yes	(null)	7	(null)
8	NUMBER_OF_EQUIPMENT_SOLD	NUMBER	Yes	(null)	8	(null)
9	NUMBER_OF_SALES	NUMBER	Yes	(null)	9	(null)

Hire Fact Table

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TIME_ID	NUMBER	Yes	(null)	1	(null)
2	SEASON_ID	NUMBER	Yes	(null)	2	(null)
3	CUSTOMER_TYPE_ID	NUMBER	Yes	(null)	3	(null)
4	CATEGORY_ID	NUMBER	Yes	(null)	4	(null)
5	COMPANY_BRANCH	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)
6	TOTAL_EQUIPMENT_HIRED_REVENUE	NUMBER	Yes	(null)	6	(null)
7	NUMBER_OF_EQUIPMENT_HIRED	NUMBER	Yes	(null)	7	(null)
8	NUMBER_OF_HIRES	NUMBER	Yes	(null)	8	(null)

B. Data Analytic Stage

Findings Report

Overall Revenue Performance

Findings:

- Total sales revenue for 2020 reached $11,583,050, marking a significant increase of $8,396,650 (263.52%) compared to the end of 2018.
- Total hire revenue for 2020 contributed $140,980, which also grew by $95,370 (209.10%) from 2018.

Evidence of Findings:

The cumulative total sales revenue was calculated using the following SQL statement:

```
SELECT
    T.TIME_ID,
        TO_CHAR(SUM(S.TOTAL_SALES_REVENUE),   '999,999,999')   AS
TOTAL_SALES_REVENUE,
       TO_CHAR(SUM(SUM(S.TOTAL_SALES_REVENUE))  OVER  (ORDER  BY
T.TIME_ID
        ROWS UNBOUNDED PRECEDING), '999,999,999') AS CUMULATIVE
FROM
    SALES_FACT S,
    TIMEDIM T
WHERE S.TIME_ID = T.TIME_ID
GROUP BY T.TIME_ID
ORDER BY T.TIME_ID;
```

Output:

All rows fetched: 33 in 0.119 seconds

	TIME_ID	TOTAL_SALES_REVENUE	CUMULATIVE
8	201811	222,000	2,860,600
9	201812	325,800	3,186,400
10	201901	178,400	3,364,800
11	201902	356,800	3,721,600
12	201903	314,200	4,035,800
13	201904	166,000	4,201,800
14	201905	547,400	4,749,200
15	201906	117,000	4,866,200
16	201907	312,000	5,178,200
17	201908	228,000	5,406,200
18	201909	226,800	5,633,000
19	201910	256,450	5,889,450
20	201911	296,600	6,186,050
21	201912	264,400	6,450,450
22	202001	356,600	6,807,050
23	202002	220,800	7,027,850
24	202003	166,500	7,194,350
25	202004	338,000	7,532,350
26	202005	454,600	7,986,950
27	202006	933,100	8,920,050
28	202007	241,500	9,161,550
29	202008	107,000	9,268,550
30	202009	646,000	9,914,550
31	202010	642,000	10,556,550
32	202011	360,100	10,916,650
33	202012	666,400	11,583,050

The cumulative total hire revenue was calculated using the following SQL statement:

```
SELECT
    T.TIME_ID,
     TO_CHAR(SUM(H.TOTAL_EQUIPMENT_HIRED_REVENUE),  '999,999,999')
AS TOTAL_EQUIPMENT_HIRED_REVENUE,
    TO_CHAR(SUM(SUM(H.TOTAL_EQUIPMENT_HIRED_REVENUE)) OVER (ORDER
BY T.TIME_ID
        ROWS UNBOUNDED PRECEDING), '999,999,999') AS CUMULATIVE
FROM
    HIRE_FACT H,
    TIMEDIM T
```

```
WHERE H.TIME_ID = T.TIME_ID
GROUP BY T.TIME_ID
ORDER BY T.TIME_ID;
```

Output:

All rows fetched: 32 in 0.247 seconds

	TIME_ID	TOTAL_EQUIPMENT_HIRED_REVENUE	CUMULATIVE
7	201811	9,430	40,235
8	201812	5,375	45,610
9	201901	3,610	49,220
10	201902	1,500	50,720
11	201903	1,980	52,700
12	201904	3,940	56,640
13	201905	6,795	63,435
14	201906	7,260	70,695
15	201907	2,255	72,950
16	201908	2,980	75,930
17	201909	3,830	79,760
18	201910	8,880	88,640
19	201911	3,220	91,860
20	201912	1,845	93,705
21	202001	1,250	94,955
22	202002	5,110	100,065
23	202003	3,080	103,145
24	202004	4,400	107,545
25	202005	5,760	113,305
26	202006	560	113,865
27	202007	5,000	118,865
28	202008	5,860	124,725
29	202009	6,625	131,350
30	202010	5,070	136,420
31	202011	3,980	140,400
32	202012	580	140,980

Number of Sales by Sales Price Scale (Low/Medium/High)

Findings:

- 97.3% of total sales revenue was generated from high sales, indicating strong demand for premium equipments.

Evidence:

The following SQL statement was used:

```
SELECT
    SA.SalesPriceScale_Desc,
    SUM(S.NUMBER_OF_SALES) AS NUMBER_OF_SALES
FROM
    SALES_FACT S,
    SALESPRICESCALEDIM SA
WHERE S.SALESPRICESCALE_ID = SA.SALESPRICESCALE_ID
GROUP BY SA.SALESPRICESCALE_DESC
ORDER BY SA.SALESPRICESCALE_DESC;
```

Output:

All rows fetched: 2 in 0.114 seconds

	SALESPRICESCALE_DESC	NUMBER_OF_SALES
1	High	146
2	Medium	4

Total Sales and Hire Revenue by Season

Findings:

- Spring was the most profitable season, generating $3,633,350 in total sales and $49,025 in equipment hire revenue.
- Winter followed closely with $3,459,600 in sales, while Summer and Autumn recorded $2,368,200 and $2,120,900, respectively.
- Total equipment hire revenue was the highest in spring, reinforcing the season's overall dominance in revenue generation.

Evidence:

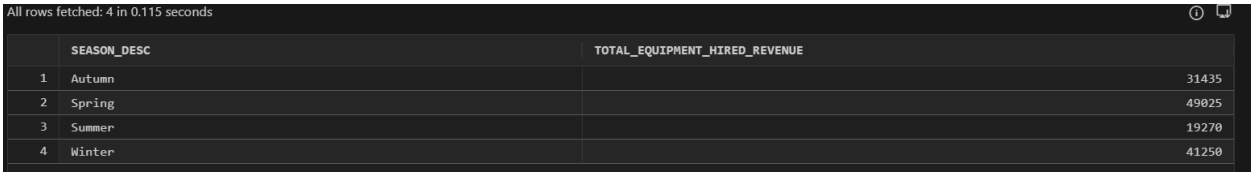
The following SQL statements were used to find the total sales revenue and total hire revenue by season:

```
SELECT
    SE.SEASON_DESC,
    SUM(S.TOTAL_SALES_REVENUE) AS TOTAL_SALES_REVENUE
FROM SALES_FACT S, SEASONDIM SE
WHERE S.SEASON_ID = SE.SEASON_ID
GROUP BY SE.SEASON_DESC
ORDER BY SE.SEASON_DESC;

SELECT
    S.SEASON_DESC,
                    SUM(H.TOTAL_EQUIPMENT_HIRED_REVENUE)         AS
TOTAL_EQUIPMENT_HIRED_REVENUE
FROM
    HIRE_FACT H,
    SEASONDIM S
WHERE H.SEASON_ID = S.SEASON_ID
GROUP BY S.SEASON_DESC
ORDER BY S.SEASON_DESC;
```

Output:

All rows fetched: 4 in 0.117 seconds

	SEASON_DESC	TOTAL_SALES_REVENUE
1	Autumn	2120900
2	Spring	3633350
3	Summer	2369200
4	Winter	3459600


All rows fetched: 4 in 0.115 seconds

	SEASON_DESC	TOTAL_EQUIPMENT_HIRED_REVENUE
1	Autumn	31435
2	Spring	49025
3	Summer	19270
4	Winter	41250

Top 5 Company Branches with the Highest Total Sales Revenue

Findings:

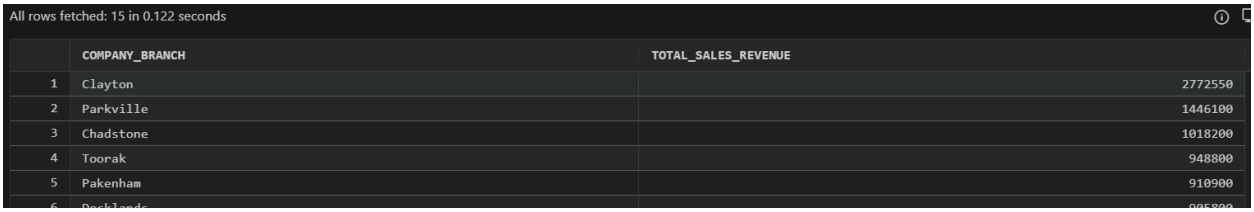
- Clayton led the highest total sales revenue, and other high-performing branches included:
  - Parkville
  - Chadstone
  - Toorak
  - Pakenham

Evidence:

The following SQL code was used:

```
SELECT
    CO.COMPANY_BRANCH,
    SUM(S.TOTAL_SALES_REVENUE) AS TOTAL_SALES_REVENUE
FROM
    SALES_FACT S,
    COMPANYBRANCHDIM CO
WHERE S.COMPANY_BRANCH = CO.COMPANY_BRANCH
GROUP BY CO.COMPANY_BRANCH
ORDER BY SUM(S.TOTAL_SALES_REVENUE) DESC;
```

Output:


All rows fetched: 15 in 0.122 seconds

	COMPANY_BRANCH	TOTAL_SALES_REVENUE
1	Clayton	2772550
2	Parkville	1446100
3	Chadstone	1018200
4	Toorak	948800
5	Pakenham	910900

Average Sales/Hire Revenue per Sale/Hire Transaction by Customer Type

Findings:

- Individual customers tend to pay more per transaction than businesses
- This indicates that individuals may opt for higher-value purchases or premium equipment compared to businesses.

Evidence:

The following SQL was used:

```
    SELECT
        C.DESCRIPTION AS CUSTOMER_TYPE,
        SUM(S.NUMBER_OF_EQUIPMENT_SOLD) AS NUMBER_OF_EQUIPMENT_SOLD,
        SUM(S.TOTAL_SALES_REVENUE) AS TOTAL_SALES_REVENUE,
        SUM(S.NUMBER_OF_SALES) AS NUMBER_OF_SALES,
           SUM(S.TOTAL_SALES_REVENUE)   /   SUM(S.NUMBER_OF_SALES)   AS
    AVG_SALES_REVENUE_PER_SALE_TRANSACTION
    FROM
        SALES_FACT S,
        CUSTOMERTYPEDIM C
    WHERE S.CUSTOMER_TYPE_ID = C.CUSTOMER_TYPE_ID
    GROUP BY C.DESCRIPTION
    ORDER BY C.DESCRIPTION;

    SELECT
        C.DESCRIPTION AS CUSTOMER_TYPE,
        SUM(H.NUMBER_OF_EQUIPMENT_HIRED) AS NUMBER_OF_EQUIPMENT_HIRED,
                       SUM(H.TOTAL_EQUIPMENT_HIRED_REVENUE)         AS
    TOTAL_EQUIPMENT_HIRED_REVENUE,
        SUM(H.NUMBER_OF_HIRES) AS NUMBER_OF_HIRES,
        SUM(H.TOTAL_EQUIPMENT_HIRED_REVENUE) / SUM(H.NUMBER_OF_HIRES)
    AS AVG_HIRE_REVENUE_PER_HIRE_TRANSACTION
    FROM
        HIRE_FACT H,
        CUSTOMERTYPEDIM C
    WHERE H.CUSTOMER_TYPE_ID = C.CUSTOMER_TYPE_ID
    GROUP BY C.DESCRIPTION
ORDER BY C.DESCRIPTION;
```

Output:

All rows fetched: 2 in 0.127 seconds

	CUSTOMER_TYPE	NUMBER_OF_EQUIPMENT_SOLD	TOTAL_SALES_REVENUE	NUMBER_OF_SALES	AVG_SALES_REVENUE_PER_SALE_TRANSACTION
1	Business	218	6423150	85	75566.4705882352941176470588235294117647
2	Individual	175	5159900	65	79383.0769230769230769230769230769230769

All rows fetched: 2 in 0.130 seconds

	CUSTOMER_TYPE	NUMBER_OF_EQUIPMENT_HIRED	TOTAL_EQUIPMENT_HIRED_REVENUE	NUMBER_OF_HIRES	AVG_HIRE_REVENUE_PER_HIRE_TRANSACTION
1	Business	320	72725	156	466.185897435897435897435897435897435897
2	Individual	297	68255	144	473.993055555555555555555555555555555556

Top 5 Most Hired Equipment Category

Findings:

- The top 5 most hired equipment categories were:
  - Trailers

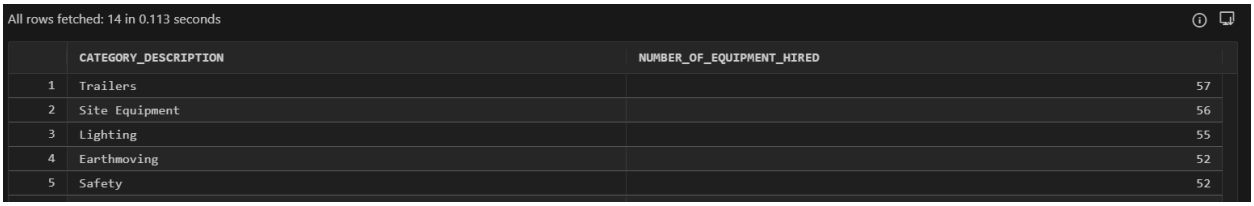
    - Site Equipment
    - Lighting
    - Earthmoving
    - Safety Equipment
- Trailers and site equipment dominated hire demand, suggesting their critical role in customer projects.

Evidence:

The following SQL statement was used:

```
SELECT
    C.CATEGORY_DESCRIPTION,
    SUM(H.NUMBER_OF_EQUIPMENT_HIRED) AS NUMBER_OF_EQUIPMENT_HIRED
FROM
    HIRE_FACT H,
    CATEGORYDIM C
WHERE H.CATEGORY_ID = C.CATEGORY_ID
GROUP BY C.CATEGORY_DESCRIPTION
ORDER BY SUM(H.NUMBER_OF_EQUIPMENT_HIRED) DESC;
```

Output:


All rows fetched: 14 in 0.113 seconds

	CATEGORY_DESCRIPTION	NUMBER_OF_EQUIPMENT_HIRED
1	Trailers	57
2	Site Equipment	56
3	Lighting	55
4	Earthmoving	52
5	Safety	52

Appendix

SQL Statements for Dimension Tables Implementation

```
-- Create TimeDIM table

DROP TABLE TimeDIM CASCADE CONSTRAINT PURGE;

CREATE TABLE TimeDIM AS
SELECT DISTINCT
    TO_CHAR(START_DATE, 'YYYYMM') AS TIME_ID,
    TO_CHAR(START_DATE, 'MM') AS TIME_MONTH,
    TO_CHAR(START_DATE, 'YYYY') AS TIME_YEAR
FROM HIRE

UNION
```

```
SELECT DISTINCT
    TO_CHAR(SALES_DATE, 'YYYYMM') AS TIME_ID,
    TO_CHAR(SALES_DATE, 'MM') AS TIME_MONTH,
    TO_CHAR(SALES_DATE, 'YYYY') AS TIME_YEAR
FROM SALES;

SELECT * FROM TimeDIM;


-- Create SeasonDIM

DROP TABLE SeasonDIM CASCADE CONSTRAINT PURGE;

CREATE TABLE SeasonDIM (
    SEASON_ID   NUMBER,
    SEASON_DESC VARCHAR2(10),
    START_MONTH NUMBER,
    END_MONTH   NUMBER
);

INSERT INTO SeasonDIM VALUES (1, 'Spring', 9, 11);

INSERT INTO SeasonDIM VALUES (2, 'Summer', 12, 2);

INSERT INTO SeasonDIM VALUES (3, 'Autumn', 3, 5);

INSERT INTO SeasonDIM VALUES (4, 'Winter', 6, 8);

SELECT * FROM SeasonDIM;


-- Create CustomerTypeDIM

DROP TABLE CustomerTypeDIM CASCADE CONSTRAINT PURGE;

CREATE TABLE CustomerTypeDIM AS
SELECT * FROM CUSTOMER_TYPE;

SELECT * FROM CustomerTypeDIM;


-- Create CategoryDIM

DROP TABLE CategoryDIM CASCADE CONSTRAINT PURGE;

CREATE TABLE CategoryDIM AS
SELECT * FROM CATEGORY;
```

```
SELECT * FROM CategoryDIM;


-- Create CompanyBranchDIM

DROP TABLE CompanyBranchDIM CASCADE CONSTRAINT PURGE;

CREATE TABLE CompanyBranchDIM AS
SELECT DISTINCT COMPANY_BRANCH
FROM STAFF;

SELECT * FROM CompanyBranchDIM;


-- Create SalesPriceScaleDIM

DROP TABLE SalesPriceScaleDIM CASCADE CONSTRAINT PURGE;

CREATE TABLE SalesPriceScaleDIM (
    SalesPriceScale_ID      NUMBER,
    SalesPriceScale_Desc    VARCHAR2(10),
    MinSalesPrice           NUMBER,
    MaxSalesPrice           NUMBER
);

INSERT INTO SalesPriceScaleDIM VALUES(1, 'Low', 0, 4999.99);

INSERT INTO SalesPriceScaleDIM VALUES(2, 'Medium', 5000, 9999.99);

INSERT INTO SalesPriceScaleDIM VALUES(3, 'High', 10000, NULL);

SELECT * FROM SalesPriceScaleDIM;
```

SQL Statements for Fact Tables Implementation

```
-- Create TEMPFACT_SALES table

DROP TABLE TEMPFACTSALES CASCADE CONSTRAINT PURGE;

CREATE TABLE TEMPFACTSALES AS
SELECT
    S.SALES_ID,
    S.SALES_DATE,
    C.CUSTOMER_TYPE_ID,
    E.CATEGORY_ID,
    ST.COMPANY_BRANCH,
    S.QUANTITY,
```

```
    S.TOTAL_SALES_PRICE
FROM
    SALES   S,
    CUSTOMER_CLEAN  C,
    EQUIPMENT   E,
    STAFF   ST
WHERE
        S.CUSTOMER_ID = C.CUSTOMER_ID
    AND S.STAFF_ID = ST.STAFF_ID
    AND S.EQUIPMENT_ID = E.EQUIPMENT_ID;

SELECT * FROM TEMPFACTSALES;

-- ADD TIME_ID INTO TEMPFACT_SALES TABLE

ALTER TABLE TEMPFACTSALES ADD (
    TIME_ID     NUMBER
);

SELECT * FROM TEMPFACTSALES;

UPDATE TEMPFACTSALES
SET TIME_ID = TO_CHAR(SALES_DATE, 'YYYYMM');

SELECT * FROM TEMPFACTSALES;

-- ADD SEASON_ID INTO TEMPFACT_SALES TABLE

ALTER TABLE TEMPFACTSALES ADD (
    SEASON_ID   NUMBER
);

UPDATE TEMPFACTSALES
SET SEASON_ID = CASE
                WHEN TO_CHAR(SALES_DATE, 'MM') >= 09
                    AND TO_CHAR(SALES_DATE, 'MM') <= 11 THEN 1
                WHEN TO_CHAR(SALES_DATE, 'MM') >= 12
                    OR TO_CHAR(SALES_DATE, 'MM') <= 2 THEN 2
                WHEN TO_CHAR(SALES_DATE, 'MM') >= 3
                    AND TO_CHAR(SALES_DATE, 'MM') <= 5 THEN 3
                WHEN TO_CHAR(SALES_DATE, 'MM') >= 6
                    AND TO_CHAR(SALES_DATE, 'MM') <= 8 THEN 4
                END;

SELECT * FROM TEMPFACTSALES;

-- ADD SALESPRICESCALE INTO TEMPFACT_SALES TABLE
```

```
ALTER TABLE TEMPFACTSALES ADD (
    SalesPriceScale_ID  NUMBER
);

SELECT * FROM TEMPFACTSALES;

UPDATE TEMPFACTSALES
SET SALESPRICESCALE_ID = CASE
                         WHEN TOTAL_SALES_PRICE < 5000 THEN 1
                         WHEN TOTAL_SALES_PRICE >= 5000
                              AND TOTAL_SALES_PRICE < 10000 THEN 2
                         WHEN TOTAL_SALES_PRICE >= 5000 THEN 3
                         END;

SELECT * FROM TEMPFACTSALES;


-- CREATE SALESFACT TABLE

DROP TABLE SALES_FACT CASCADE CONSTRAINT PURGE;

CREATE TABLE SALES_FACT AS
    SELECT
        TIME_ID,
        SEASON_ID,
        CUSTOMER_TYPE_ID,
        CATEGORY_ID,
        COMPANY_BRANCH,
        SALESPRICESCALE_ID,
        SUM(TOTAL_SALES_PRICE) AS TOTAL_SALES_REVENUE,
        SUM(QUANTITY) AS NUMBER_OF_EQUIPMENT_SOLD,
        COUNT(SALES_ID) AS NUMBER_OF_SALES
    FROM
        TEMPFACTSALES
    GROUP BY
        TIME_ID,
        SEASON_ID,
        CUSTOMER_TYPE_ID,
        CATEGORY_ID,
        COMPANY_BRANCH,
        SALESPRICESCALE_ID
    ORDER BY
        TIME_ID;

SELECT * FROM SALES_FACT;


-- CREATE TEMPFACTHIRE TABLE
```

```
DROP TABLE TEMPFACTHIRE CASCADE CONSTRAINT PURGE;

CREATE TABLE TEMPFACTHIRE AS
SELECT
    H.HIRE_ID,
    H.START_DATE,
    H.END_DATE,
    ST.COMPANY_BRANCH,
    C.CUSTOMER_TYPE_ID,
    E.CATEGORY_ID,
    H.QUANTITY,
    H.TOTAL_HIRE_PRICE
FROM
    HIRE    H,
    STAFF   ST,
    CUSTOMER_CLEAN  C,
    EQUIPMENT   E
WHERE
        H.CUSTOMER_ID = C.CUSTOMER_ID
    AND H.STAFF_ID = ST.STAFF_ID
    AND H.EQUIPMENT_ID = E.EQUIPMENT_ID;

SELECT * FROM TEMPFACTHIRE;

-- ADD TIME_ID INTO TEMPFACTHIRE

ALTER TABLE TEMPFACTHIRE ADD (
    TIME_ID     NUMBER
);

SELECT * FROM TEMPFACTHIRE;

UPDATE TEMPFACTHIRE
SET TIME_ID = TO_CHAR(START_DATE, 'YYYYMM');

SELECT * FROM TEMPFACTHIRE;

-- ADD SEASON_ID INTO TEMPFACTHIRE

ALTER TABLE TEMPFACTHIRE ADD (
    SEASON_ID   NUMBER
);

SELECT * FROM TEMPFACTHIRE;

UPDATE TEMPFACTHIRE
SET SEASON_ID = CASE
```

```
                WHEN TO_CHAR(START_DATE, 'MM') >= 09
                    AND TO_CHAR(START_DATE, 'MM') <= 11 THEN 1
                WHEN TO_CHAR(START_DATE, 'MM') >= 12
                    OR TO_CHAR(START_DATE, 'MM') <= 2 THEN 2
                WHEN TO_CHAR(START_DATE, 'MM') >= 3
                    AND TO_CHAR(START_DATE, 'MM') <= 5 THEN 3
                WHEN TO_CHAR(START_DATE, 'MM') >= 6
                    AND TO_CHAR(START_DATE, 'MM') <= 8 THEN 4
                END;

SELECT * FROM TEMPFACTHIRE;


-- CREATE HIRE_FACT TABLE

DROP TABLE HIRE_FACT CASCADE CONSTRAINT PURGE;

CREATE TABLE HIRE_FACT AS
SELECT
    TIME_ID,
    SEASON_ID,
    CUSTOMER_TYPE_ID,
    CATEGORY_ID,
    COMPANY_BRANCH,
    SUM(TOTAL_HIRE_PRICE) AS TOTAL_EQUIPMENT_HIRED_REVENUE,
    SUM(QUANTITY) AS NUMBER_OF_EQUIPMENT_HIRED,
    COUNT(HIRE_ID) AS NUMBER_OF_HIRES
FROM TEMPFACTHIRE
GROUP BY
    TIME_ID,
    SEASON_ID,
    CUSTOMER_TYPE_ID,
    CATEGORY_ID,
    COMPANY_BRANCH
ORDER BY
    TIME_ID;

SELECT * FROM HIRE_FACT;
```