MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
28901797	NASSAR	AHMED MOHAMED H A
30241510	NATHANIA	GRACE
29764920	KAN	KELVIN JIA YAW

* Please include the names of all other group members.

Unit name and code	FIT3003: BUSINESS INTELLIGENCE AND DATA WAREHOUSING	
Title of assignment	MAJOR ASSIGNMENT: MONCHEF	
Lecturer/tutor	DR. SOON LAY KI	
Tutorial day and time	THURSDAY 10 AM – 12 PM	**Campus** MALAYSIA
Is this an authorised group assignment?	☑ **Yes** ☐ **No**	
Has any part of this assignment been previously submitted as part of another unit/course?	☐ **Yes** ☑ **No**	
Due Date FRIDAY NOVEMBER 6 2020	**Date submitted** WEDNESDAY NOVEMBER 4 2020	

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date) ..... **Signature of lecturer/tutor .....**

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature ..... *Date* 4 November 2020

* delete (iii) if not applicable

Signature _____ Date: 4 November 2020 Signature _____ Date: _____

Signature _____ Date: 4 November 2020 Signature _____ Date: _____

Signature _____ Date: 4 November 2020 Signature _____ Date: _____

Privacy Statement

The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

Contribution Declaration Form
(to be completed by all team members)

Please fill in the form with the contribution from each student towards the assignment.

1 NAME AND CONTRIBUTION DETAILS

Student ID	Student Name	Contribution Percentage
28901797	AHMED MOHAMED H A NASSAR	33.3%
30241510	GRACE NATHANIA	33.3%
29764920	KELVIN KAN JIA YAW	33.3%

AHMED MOHAMED H A NASSAR	1. Data Cleaning Documentation and Proofreading 2. ERD Design 3. Star Schema Design 4. Star Schema Implementation: Dimensions for Level 0 and Level 2 5. Reports Testing 6. Business Intelligence Dashboard Design
GRACE NATHANIA	1. Data Cleaning 2. Data Cleaning Documentation 3. Star Schema Design 4. Star Schema Implementation : Facts Level 0 5. Reports Task 1, 3 6. Generation of Graphs
KEVLIN KAN JIA YAW	1. Data Cleaning 2. Data Cleaning Documentation 3. ERD refinement and proofreading 4. Star Schema Design 5. Star Schema Implementation : Facts Level 2 6. Reports Task 2, 4

2 DECLARATION

We declare that:

- The information we have supplied in or with this form is complete and correct.
- We understand that the information we have provided in this form will be used for individual assessment of the assignment.

3 SIGNATURE

Signatures

Day Month Year

Date 4 / 11 / 2020

MONASH University

MonChef DataWarehouse

FIT3003 Major Assignment
Semester 2, 2020

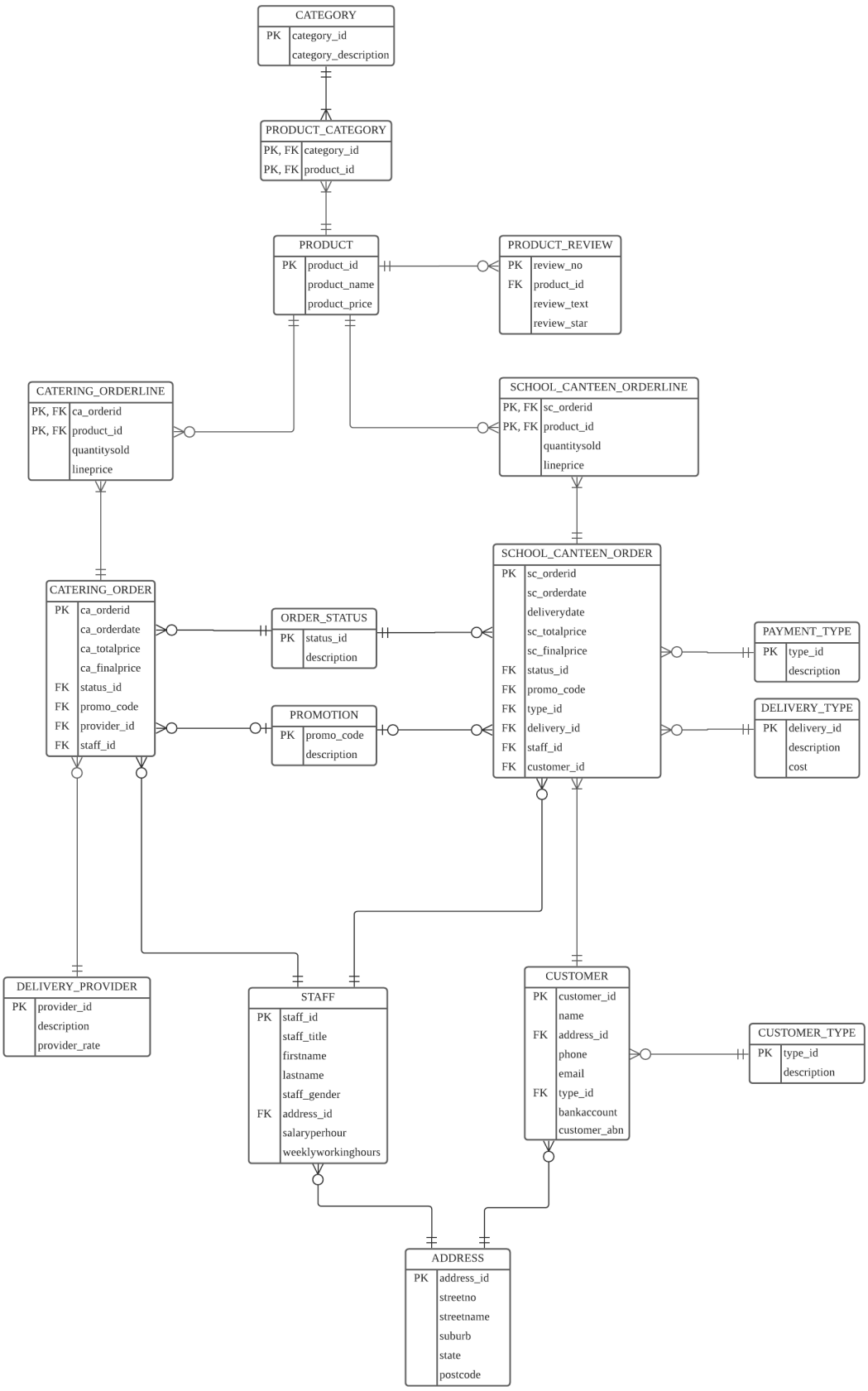
Ahmed Nassar – 28901797
Grace Nathania - 30241510
Kelvin Kan Jia Yaw – 29764920

Submitted on Wednesday November 4 2020

Table of Contents

Operational Database

ERD Diagram



Data Cleaning Methods

If the table is small, we will manually select the preview of the whole contents and records of the table to look out for any dirty data.

If not, we will use "**desc tableName**" to see which column does not have a "Not Null" constraint to determine that for that column, there should exist a value.

If a column does not have a "not null constraint" when it is supposed to have one, we will check for any null value in that column.

Then, we will check for duplicates records by using count(*) and count(distinct x), where x is the primary key.

After that, if a table consists of a foreign key, we will check if the foreign key actually exist in the parent table by using "**select count (FK) from TableName where FK not in (select FK from ParentTableName);**".

Other than that, if a table consists of numeric data/date, we will check whether the numeric data/date (e.g address_streetno, address_postcode, price, quantity ordered, order date, staff_salaryperhour) recorded is sensible. For example, price, quantity ordered and staff_salary per hour should not be of a negative value.
SQL statement format for checking if there is any record of sc_totalprice/sc_finalpricein school_canteen_order is lesser than 0: "**select * from MonChef.school_canteen_order where sc_totalprice < 0 or sc_finalprice < 0;**"

An example of an insensible date data would be a month being less than 01 or higher than 12. Another example would be if a day is less than 01 or higher than 31
The sql statement format would be:
"**select distinct to_char(sc_orderdate, 'YYYY') from MonChef.school_canteen_order;**
select distinct to_char(sc_orderdate, 'MM') as MM from MonChef.school_canteen_order order by MM;
select distinct to_char(sc_orderdate, 'DD') as DD from MonChef.school_canteen_order order by DD;"

Regarding the school_canteen_order table, we check if there is any delivery date that is earlier than the order date as this would not make sense.
The SQL statement format for that is "**select * from MonChef.school_canteen_order where sc_deliverydate < sc_orderdate;**".

List of Errors

With the strategy described above, the team has found a total of 10 errors or “dirty” data in the operational database, listed below.

1. **null value** in category_id (table: category)
2. **duplicate** data record (table: product)
3. **null value** in product_name and product_price (table: product)
4. there is a staff with **negative salary and working hours**, **address_id is null** for staff_id: ST051 (table: staff)
5. **duplicate** data record (table: customer)
6. **A delivery date took place before the sc_orderdate** for SC_ORDERID: S1905 and sc_orderid: S1906 (table: school_canteen_order)
7. **type_id= CU035 is invalid foreign key as it does not exist in parent table** for sc_orderid: S1906. (table: school_canteen_order)
8. **Duplicate** data value ca_order resulting in unique value violation (table: catering_order)
9. **Date that is higher than the current date** ca_orderdate: “ 2035” (table: catering_order)
10. ca_orderid: C1890 and productid: F0199 **not valid foreign key** as they don't exist in parent table (table: catering_orderline)
11. **product_id : F0198 is invalid foreign key as it does not exist in parent table** for review_no: R1606. (table: product_review)

Error [1]

null value in category_id (table: category)

Detect Error SQL

```
desc monchef.category;
SELECT *
FROM   monchef.category;
```

Output

	CATEGORY_ID	CATEGORY_DESCRIPTION
1	FC001	Savoury
2	FC002	Sweet
3	FC003	Starter
4	FC004	Main
5	FC005	Dessert
6	FC006	Drink
7	FC007	Indonesian
8	FC008	Korean
9	FC009	Thai
10	(null)	Unknown

Fix Error SQL

```
CREATE TABLE category AS
  SELECT *
  FROM   monchef.category
  WHERE  category_id IS NOT NULL
         AND category_description != 'Unknown';
```

Output

	CATEGORY_ID	CATEGORY_DESCRIPTION
1	FC001	Savoury
2	FC002	Sweet
3	FC003	Starter
4	FC004	Main
5	FC005	Dessert
6	FC006	Drink
7	FC007	Indonesian
8	FC008	Korean
9	FC009	Thai

Error [2] & [3]

Duplicate data and null value in product_name and product_price (table: product)

Detect Error SQL

```
-- check if there is any duplicate data
SELECT Count (DISTINCT product_id)
FROM   monchef.product; -- 191

SELECT Count (*)
FROM   monchef.product; -- 195: duplicate exist

-- check if there is any null value (null exists in product_name and
product_price)
desc monchef.product;
SELECT DISTINCT *
FROM   monchef.product
WHERE  product_name IS NULL
        OR product_price IS NULL
ORDER  BY product_id;
```

Output

187	F0187	Donat Coklat Keju	10
188	F0188	Risoles Ragout Ayam	11
189	F0189	Siomay Ayam Udang	11
190	F0190	Pukis	10
191	F0179	Iga Bakar (Grilled Beef Ribs)	14
192	F0179	Iga Bakar (Grilled Beef Ribs)	14
193	F0179	Iga Bakar (Grilled Beef Ribs)	14
194	F0056	Pad Thai	16.9
195	F0195	(null)	(null)

Fix Error SQL

```
CREATE TABLE product AS
  SELECT DISTINCT *
  FROM   monchef.product
  WHERE  product_name IS NOT NULL
         AND product_price IS NOT NULL
  ORDER  BY product_id;
```

Output

178	F0178	Ayam Bakar (Grilled Chicken)	14
179	F0179	Iga Bakar (Grilled Beef Ribs)	14
180	F0180	Gado Gado (V)	11
181	F0181	Ayam Geprek	12
182	F0182	Sate Ayam Madura	12
183	F0183	Ayam Bawang Sambal Matah	20
184	F0184	Martabak Manis	10
185	F0185	Bakso Goreng	10
186	F0186	Kue Sus	10
187	F0187	Donat Coklat Keju	10
188	F0188	Risoles Ragout Ayam	11
189	F0189	Siomay Ayam Udang	11
190	F0190	Pukis	10

Error [4]

There is a staff with negative salary and working hours, address_id also does not exist for STAFF_ID: ST051 (table: staff)

Detect Error SQL

```
SELECT *
FROM   monchef.staff
WHERE  staff_salaryperhour < 0
        OR staff_weeklyworkinghours < 0; -- ST051
-- check if address_id exists
SELECT *
FROM   monchef.staff
WHERE  address_id NOT IN (SELECT DISTINCT address_id
                          FROM   address);
```

Output

45	ST045	Mrs	Ebony	Dallachy	F	25.25	24	AD094
46	ST046	Ms	Brooke	Cordeaux	F	25.25	24	AD095
47	ST047	Mr	Zac	Cran	M	25.25	27	AD096
48	ST048	Mrs	Amy	Eardley-Wilmot	F	30	29	AD097
49	ST049	Mr	Seth	Beal	M	30	30	AD098
50	ST050	Mr	Toby	Greaves	M	20.5	29	AD099
51	ST051	Mr	Arnold	Baker	M	-50	-1	AD999

Fix Error SQL

```
CREATE TABLE staff AS
  SELECT *
  FROM   monchef.staff
  WHERE  staff_id != 'ST051';
```

Output

44	ST044	Mrs	Natalie	Appel	F	30	23	AD093
45	ST045	Mrs	Ebony	Dallachy	F	25.25	24	AD094
46	ST046	Ms	Brooke	Cordeaux	F	25.25	24	AD095
47	ST047	Mr	Zac	Cran	M	25.25	27	AD096
48	ST048	Mrs	Amy	Eardley-Wilmot	F	30	29	AD097
49	ST049	Mr	Seth	Beal	M	30	30	AD098
50	ST050	Mr	Toby	Greaves	M	20.5	29	AD099

Error [5]

Duplicated data (customer)

Detect Error SQL

```
-- check if there is any duplicate value
SELECT Count(*)
FROM   monchef.customer; -- 57

SELECT Count (DISTINCT customer_id)
FROM   monchef.customer; -- 50  duplicate exists
```

Output

46	CU046	Olivia Hytten	0474942370	olivia.hytten@gmail.com	034556-71985042	72437636412	CT003	AD046
47	CU047	Stephanie Bindon	0439334129	stephanie.bindon@gmail.com	065292-15138547	16187883245	CT002	AD047
48	CU048	Mackenzie Flinders	0424971294	mackenzie.flinders@gmail.com	014716-80643331	94617327539	CT002	AD048
49	CU049	Michael Andrew	0474995797	michael.andrew@gmail.com	034716-24777494	13340820116	CT003	AD049
50	CU050	Brodie Heinig	0424934968	brodie.heinig@gmail.com	084485-66356527	89907363765	CT003	AD050
51	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
52	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
53	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
54	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
55	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
56	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
57	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036

Fix Error SQL

```
CREATE TABLE customer AS
  SELECT DISTINCT *
  FROM   monchef.customer;
```

Output

32	CU032	Jackson McConnan	0436293182	jackson.mcconnan@gmail.com	015299-84577139	null	CT001	AD032
33	CU033	Lola Keast	0439320648	lola.keast@gmail.com	015521-36812475	84704169155	CT003	AD033
34	CU034	Kayla Goddard	0489480000	kayla.goddard@gmail.com	015558-12766024	52902477959	CT002	AD034
35	CU035	Liam Clement	0429251717	liam.clement@gmail.com	015239-96838931	75002611517	CT003	AD035
36	CU036	Lara Degotardi	0424035737	lara.degotardi@gmail.com	065239-91782832	53107981253	CT002	AD036
37	CU037	Brooke Proctor	0424076525	brooke.proctor@gmail.com	084916-84717855	14598633355	CT003	AD037
38	CU038	Oscar Thirkell	0488791569	oscar.thirkell@gmail.com	035395-70685639	55322501712	CT003	AD038
39	CU039	Ryan Rubin	0435373817	ryan.rubin@gmail.com	084556-10128995	null	CT001	AD039

Error[6] & [7]

A delivery date took place before the sc_orderdate for SC_ORDERID: S1905 and sc_orderid: S1906 (table: school_canteen_order)

type_id is invalid foreign key as it does not exist in parent table for sc_orderid: S1906. (table: school_canteen_order)

Detect Error SQL

```
-- check if there is any delivery date that is smaller than order date
SELECT *
FROM   monchef.school_canteen_order
WHERE  sc_deliverydate < sc_orderdate;
-- S1905: orderdate: 12-05-2019, deliverydate: 12-04-2019
-- S1906: orderdate: 02-05-2021, deliverydate: 02-05-2019

-- check if there is any payment type_id that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  type_id NOT IN (SELECT type_id
                       FROM   payment_type); -- S1906
```

Output

1497	S1497	17/01/2019	18/01/2019	198.2	208.29	OS005	PR001	PT003	DT004	ST040	CU027
1498	S1498	02/08/2020	02/08/2020	12	12	OS005	(null)	PT004	DT002	ST015	CU027
1499	S1499	17/08/2019	17/08/2019	53.8	61.11	OS005	PR001	PT003	DT003	ST026	CU016
1500	S1500	02/05/2020	02/05/2020	356.6	376.6	OS005	(null)	PT003	DT004	ST017	CU030
1501	S1905	12/05/2019	12/04/2019	9	9	OS005	(null)	PT004	DT002	ST030	CU030
1502	S1906	02/05/2021	02/05/2019	9	9	OS006	(null)	PT014	DT001	ST030	CU035

Fix Error SQL

```
CREATE TABLE school_canteen_order AS
  SELECT *
  FROM   monchef.school_canteen_order
  WHERE  sc_orderid NOT IN ( 'S1905', 'S1906' );
```

Output

1494	S1494	08/06/2020	08/06/2020	62.7	56.43	OS005	PR002	PT002	DT002	ST031	CU021
1495	S1495	04/12/2019	05/12/2019	196.1	206.3	OS005	PR001	PT004	DT004	ST012	CU047
1496	S1496	10/04/2019	10/04/2019	64	60.8	OS005	PR001	PT003	DT001	ST027	CU046
1497	S1497	17/01/2019	18/01/2019	198.2	208.29	OS005	PR001	PT003	DT004	ST040	CU027
1498	S1498	02/08/2020	02/08/2020	12	12	OS005	(null)	PT004	DT002	ST015	CU027
1499	S1499	17/08/2019	17/08/2019	53.8	61.11	OS005	PR001	PT003	DT003	ST026	CU016
1500	S1500	02/05/2020	02/05/2020	356.6	376.6	OS005	(null)	PT003	DT004	ST017	CU030

Error [8] & [9]
duplicate ca_order & there is an order from year 2035 (catering_order)

Detect Error SQL

```
-- check if there is any duplicate data
SELECT Count(*)
FROM   monchef.catering_order; -- 1504

SELECT Count (DISTINCT ca_orderid)
FROM   monchef.catering_order; -- 1500 duplicate exis

-- check if there is date that does not make sense
SELECT DISTINCT To_char(ca_orderdate, 'YYYY')
FROM   monchef.catering_order; -- there is year 2035 which does not make sense
```

Output

	CA_ORDERID	CA_ORDERDATE	CA_TOTALPRICE	CA_FINALPRICE	STATUS_ID	PROMO_CODE	PROVIDER_ID	STAFF_ID
1	C0001	01/12/2018	7.9	7.9	OS005	(null)	DP001	ST032
2	C0001	12/05/2035	7.9	7.9	OS005	(null)	DP001	ST032
3	C0002	11/07/2020	60	67.2	OS005	(null)	DP004	ST001
4	C0003	24/04/2019	174.9	183.65	OS006	PR001	DP003	ST007
5	C0004	11/01/2020	110.7	118.45	OS005	PR001	DP004	ST009

Fix Error SQL

```
CREATE TABLE catering_order AS
  SELECT DISTINCT *
  FROM   monchef.catering_order
  WHERE  To_char(ca_orderdate, 'YYYY') != '2035';
```

Output

	CA_ORDERID	CA_ORDERDATE	CA_TOTALPRICE	CA_FINALPRICE	STATUS_ID	PROMO_CODE	PROVIDER_ID	STAFF_ID
1	C0001	01/12/2018	7.9	7.9	OS005	(null)	DP001	ST032
2	C0002	11/07/2020	60	67.2	OS005	(null)	DP004	ST001
3	C0003	24/04/2019	174.9	183.65	OS006	PR001	DP003	ST007
4	C0004	11/01/2020	110.7	118.45	OS005	PR001	DP004	ST009
5	C0005	19/01/2018	74.4	78.12	OS005	PR002	DP002	ST043

Error [10]

CA_ORDERID: C1890 does not exist as well as the ordered product (catering_orderline)

Detect Error SQL

```
-- check if there is order that does not exist
SELECT *
FROM   monchef.catering_orderline
WHERE  ca_orderid NOT IN (SELECT ca_orderid
                          FROM   catering_order); -- CA_ORDERID: C1890
```

Output

	CA_ORDERID	PRODUCT_ID	COL_QUANTITYSOLD	COL_LINEPRICE
1	C1890	F0199	3	18
2	C1500	F0008	5	22.5
3	C1500	F0106	2	35.8
4	C1499	F0176	4	480
5	C1498	F0114	3	44.7
6	C1498	F0021	5	50
7	C1498	F0060	3	50.7

Fix Error SQL

```
CREATE TABLE catering_orderline AS
  SELECT *
  FROM   monchef.catering_orderline
  WHERE  ca_orderid != 'C1890';
```

Output

	CA_ORDERID	PRODUCT_ID	COL_QUANTITYSOLD	COL_LINEPRICE
1	C1500	F0008	5	22.5
2	C1500	F0106	2	35.8
3	C1499	F0176	4	480
4	C1498	F0114	3	44.7
5	C1498	F0021	5	50
6	C1498	F0060	3	50.7
7	C1498	F0136	4	87.6

Error[11]

product_id : F0198 is invalid foreign key as it does not exist in parent table for review_no: R1606. (table: product_review)

Detect Error SQL

```
SELECT product_id
FROM   monchef.product_review
WHERE  product_id NOT IN (SELECT product_id
                          FROM   product); -- R1606
```

Output

1603	R1603	we were in Melbourne for 5 nights. I had read about...	4	F0083
1604	R1604	what a place nice location and friendly atmosphere ...	4	F0012
1605	R1605	without rice to soak up all those flavours, it all ...	3	F0025
1606	R1606	super location, awesome staff, our team absolutely ...	5	F0198

Fix Error SQL

```
CREATE TABLE product_review AS
  SELECT *
  FROM   monchef.product_review
  WHERE  product_id IN (SELECT product_id
                        FROM   product);
```

Output

1600	R1600	vibrant happy place , Food superb , Co...	5	F0091
1601	R1601	we had three courses for $39 and it wa...	4	F0021
1602	R1602	we have been to a Rice paper scissors ...	5	F0177
1603	R1603	we were in Melbourne for 5 nights. I h...	4	F0083
1604	R1604	what a place nice location and friendl...	4	F0012
1605	R1605	without rice to soak up all those flav...	3	F0025

Note

The SQL shown here is only for the errors detected and fixed. The full SQL for surveying the data and checking for any potential errors, and fixing them can be found in the next page.

Data Observation and Cleaning Full SQL

```
--drop tables commands
DROP TABLE category CASCADE CONSTRAINTS;
DROP TABLE product_category CASCADE CONSTRAINTS;
DROP TABLE product CASCADE CONSTRAINTS;
DROP TABLE product_review CASCADE CONSTRAINTS;
DROP TABLE school_canteen_orderline CASCADE CONSTRAINTS;
DROP TABLE catering_orderline CASCADE CONSTRAINTS;
DROP TABLE catering_order CASCADE CONSTRAINTS;
DROP TABLE delivery_type CASCADE CONSTRAINTS;
DROP TABLE school_canteen_order CASCADE CONSTRAINTS;
DROP TABLE promotion CASCADE CONSTRAINTS;
DROP TABLE order_status CASCADE CONSTRAINTS;
DROP TABLE payment_type CASCADE CONSTRAINTS;
DROP TABLE delivery_provider CASCADE CONSTRAINTS;
DROP TABLE staff CASCADE CONSTRAINTS;
DROP TABLE customer_type CASCADE CONSTRAINTS;
DROP TABLE address CASCADE CONSTRAINTS;
DROP TABLE customer CASCADE CONSTRAINTS;

--creating local tables
---------------- category (1 Error: null value in category_id) -----------------
-- select the whole table to see if there is an error (the table is small)
desc monchef.category;
SELECT *
FROM   monchef.category;

CREATE TABLE category AS
  SELECT *
  FROM   monchef.category
  WHERE  category_id IS NOT NULL
         AND category_description != 'Unknown';

---------------------------------- product_category (OK) ----------------------
desc monchef.product_category;
-- check if any duplicates records
SELECT product_id,
       category_id,
       Count(*)
FROM   monchef.product_category
GROUP  BY product_id,
          category_id
HAVING Count(*) > 1;

CREATE TABLE product_category AS
  SELECT *
  FROM   monchef.product_category;

---------------------------------- product (2 Errors) -------------------------
SELECT *
FROM   monchef.product;
```

```
-- check if there is any duplicate data
SELECT Count (DISTINCT product_id)
FROM   monchef.product; -- 191

SELECT Count (*)
FROM   monchef.product; -- 195: duplicate exist

SELECT product_id,
       product_name,
       product_price,
       Count(*)
FROM   monchef.product
GROUP  BY product_id,
          product_name,
          product_price
HAVING Count(*) > 1;

-- check if there is any null value (null exists in product_name and
product_price)
desc monchef.product;
SELECT DISTINCT *
FROM   monchef.product
WHERE  product_name IS NULL
        OR product_price IS NULL
ORDER  BY product_id;

-- check if any product_price is < 0
SELECT DISTINCT *
FROM   monchef.product
WHERE  product_price < 0;

CREATE TABLE product AS
  SELECT DISTINCT *
  FROM   monchef.product
  WHERE  product_name IS NOT NULL
         AND product_price IS NOT NULL
  ORDER  BY product_id;

---------------------------- product review (1 Error) ----------------------------
----
-- check if there is duplicate review
SELECT Count (*)
FROM   monchef.product_review; -- 1606

SELECT Count (DISTINCT review_no)
FROM   monchef.product_review; -- 1606
-- check if there is any null value (all comlumn consist of not null constraint)
desc monchef.product_review;
-- check if there is ordered product that does not exist
SELECT product_id
```

```
FROM   monchef.product_review
WHERE  product_id NOT IN (SELECT product_id
                          FROM   product); -- R1606
CREATE TABLE product_review AS
  SELECT *
  FROM   monchef.product_review
  WHERE  product_id IN (SELECT product_id
                        FROM   product);

---------------------------------- delivery_type (OK) --------------------------
SELECT *
FROM   monchef.delivery_type;

-- creating table
CREATE TABLE delivery_type AS
  SELECT *
  FROM   monchef.delivery_type;

---------------------------------- promotion (OK) ------------------------------
SELECT *
FROM   monchef.promotion;

-- creating table
CREATE TABLE promotion AS
  SELECT *
  FROM   monchef.promotion;

---------------------------------- order status (OK) ---------------------------
SELECT *
FROM   monchef.order_status;

-- creating table
CREATE TABLE order_status AS
  SELECT *
  FROM   monchef.order_status;

---------------------------------- payment_type (OK) ---------------------------
SELECT *
FROM   monchef.payment_type;

-- creating table
CREATE TABLE payment_type AS
  SELECT *
  FROM   monchef.payment_type;

---------------------------------- delivery_provider (OK) ----------------------
SELECT *
FROM   monchef.delivery_provider;

-- creating table
CREATE TABLE delivery_provider AS
```

```
  SELECT *
  FROM   monchef.delivery_provider;

---------------------------------- address (OK) -------------------------------
SELECT *
FROM   monchef.address;

-- check if there is any null value
desc monchef.address;
-- check if there is any duplicate data
SELECT Count(*)
FROM   monchef.address; -- 249

SELECT Count (DISTINCT address_id)
FROM   monchef.address; -- 249
-- check if there is any street number or postcode that does not make sense
SELECT *
FROM   monchef.address
WHERE  address_streetno <= 0
        OR address_postcode <= 0;

-- creating table
CREATE TABLE address AS
  SELECT *
  FROM   monchef.address;

---------------------------------- staff (1 ERROR) ----------------------------
----
SELECT *
FROM   monchef.staff;

-- check if there is any null value
desc monchef.staff;
-- check if there is any staff_title and staff_gender that do not align
SELECT *
FROM   monchef.staff
WHERE  staff_title = 'Mr'
       AND staff_gender = 'F';

SELECT *
FROM   monchef.staff
WHERE  staff_title = 'Mrs'
       AND staff_gender = 'M';

-- check if there is any duplicate data
SELECT Count(*)
FROM   monchef.staff; -- 51

SELECT Count (DISTINCT staff_id)
FROM   monchef.staff; -- 51
```

```
-- check if there is any staff_salaryperhour and staff_weeklyworkinghours that
are < 0
SELECT *
FROM   monchef.staff
WHERE  staff_salaryperhour < 0
        OR staff_weeklyworkinghours < 0; -- ST051
-- check if address_id exists
SELECT *
FROM   monchef.staff
WHERE  address_id NOT IN (SELECT DISTINCT address_id
                          FROM   address); -- ST051
-- creating table
CREATE TABLE staff AS
  SELECT *
  FROM   monchef.staff
  WHERE  staff_id != 'ST051';

---------------------------------- customer_type (OK) --------------------------
-------
SELECT *
FROM   monchef.customer_type;

-- creating table
CREATE TABLE customer_type AS
  SELECT *
  FROM   monchef.customer_type;

---------------------------------- customer (1 ERROR) --------------------------
-------
SELECT *
FROM   monchef.customer;

-- check if there is any null value
desc monchef.customer;
-- check if there is any duplicate value
SELECT Count(*)
FROM   monchef.customer; -- 57

SELECT Count (DISTINCT customer_id)
FROM   monchef.customer; -- 50  duplicate exists
-- check if there is any customer type_id that does not exist
SELECT *
FROM   monchef.customer
WHERE  type_id NOT IN (SELECT type_id
                       FROM   customer_type);

-- check if there is any adress that does not exist
SELECT *
FROM   monchef.customer
WHERE  address_id NOT IN (SELECT DISTINCT address_id
                          FROM   address);
```

```
-- creating table
CREATE TABLE customer AS
  SELECT DISTINCT *
  FROM   monchef.customer;

------------------------ school_canteen_order (2 Errors) ----------------------
SELECT *
FROM   monchef.school_canteen_order;

-- check if there is any null value (all comlumn consist of not null constraint
except promo_code)
desc monchef.school_canteen_order;
-- check if there is any duplicate value
SELECT Count(*)
FROM   monchef.school_canteen_order; -- 1502

SELECT Count(DISTINCT sc_orderid)
FROM   monchef.school_canteen_order; --1502
-- check if delivery_type exists
SELECT *
FROM   monchef.school_canteen_order
WHERE  delivery_id NOT IN (SELECT delivery_id
                           FROM   delivery_type);

-- check if there is any order date that does not make sense
SELECT DISTINCT To_char(sc_orderdate, 'YYYY')
FROM   monchef.school_canteen_order; -- there is year 2021 which does not make
sense

SELECT DISTINCT To_char(sc_orderdate, 'MM') AS MM
FROM   monchef.school_canteen_order
ORDER  BY mm;

SELECT DISTINCT To_char(sc_orderdate, 'DD') AS DD
FROM   monchef.school_canteen_order
ORDER  BY dd;

-- check if there is any delivery date that does not make sense
SELECT DISTINCT To_char(sc_deliverydate, 'YYYY')
FROM   monchef.school_canteen_order;

SELECT DISTINCT To_char(sc_deliverydate, 'MM') AS MM
FROM   monchef.school_canteen_order
ORDER  BY mm;

SELECT DISTINCT To_char(sc_deliverydate, 'DD') AS DD
FROM   monchef.school_canteen_order
ORDER  BY dd;

-- check if there is any delivery date that is smaller than order date
```

```
-- S1905: orderdate: 12-05-2019, deliverydate: 12-04-2019
-- S1906: orderdate: 02-05-2021, deliverydate: 02-05-2019
SELECT *
FROM   monchef.school_canteen_order
WHERE  sc_deliverydate < sc_orderdate;

-- check if there is any status_id that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  status_id NOT IN (SELECT status_id
                         FROM   order_status);

-- check if there is any promo_code that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  promo_code NOT IN (SELECT promo_code
                          FROM   promotion);

-- check if there is any payment type_id that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  type_id NOT IN (SELECT type_id
                       FROM   payment_type); -- S1906
-- check if there is any delivery_id that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  delivery_id NOT IN (SELECT delivery_id
                           FROM   delivery_type);

-- check if there is any staff_id that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  staff_id NOT IN (SELECT staff_id
                        FROM   staff);

-- check if there is any customer_id that does not exist
SELECT *
FROM   monchef.school_canteen_order
WHERE  customer_id NOT IN (SELECT customer_id
                           FROM   customer);

-- check if there is any price that doesnt make sense
SELECT *
FROM   monchef.school_canteen_order
WHERE  sc_totalprice < 0
        OR sc_finalprice < 0;

-- creating the table
CREATE TABLE school_canteen_order AS
  SELECT *
  FROM   monchef.school_canteen_order
```

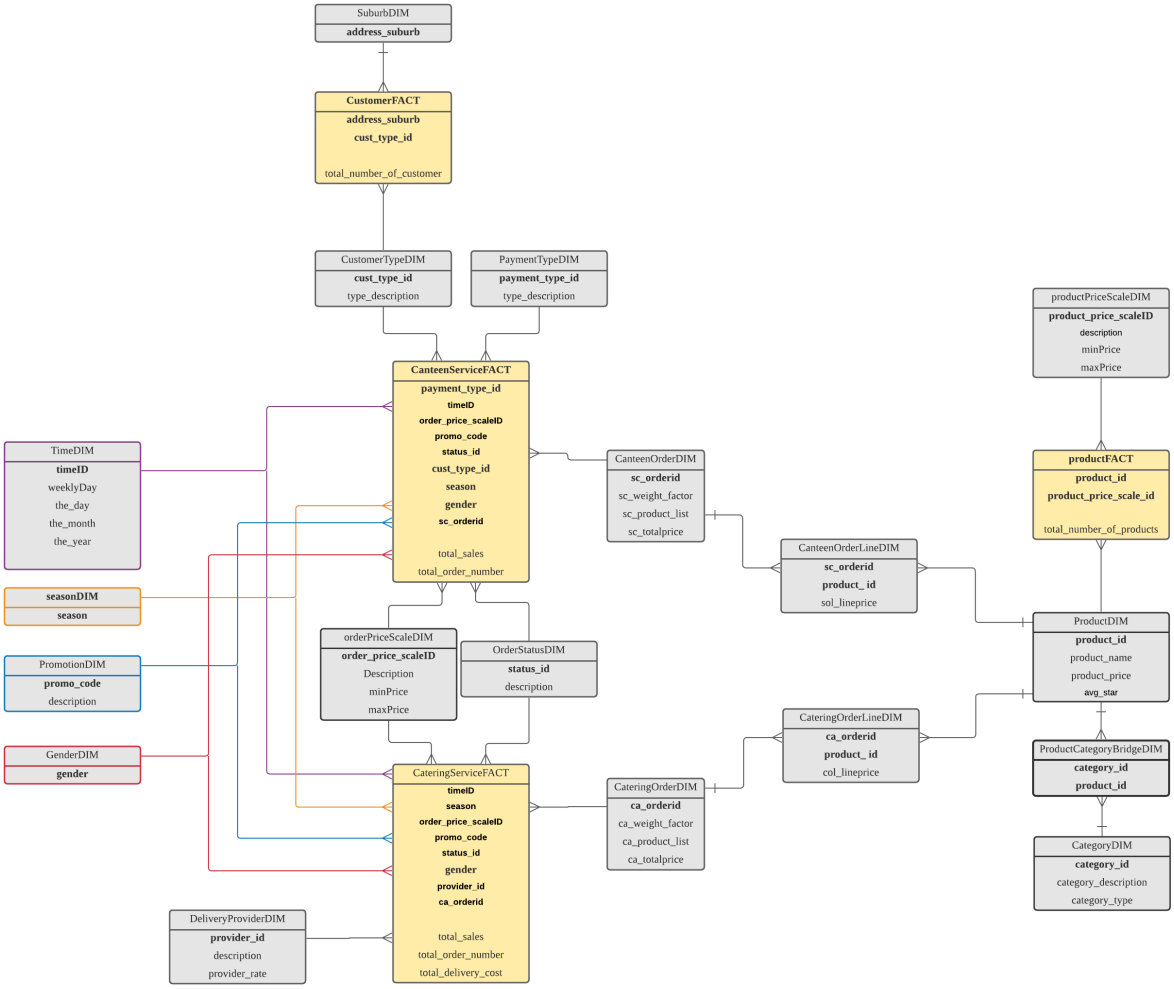
```
  WHERE  sc_orderid NOT IN ( 'S1905', 'S1906' );

---------------------- school_canteen_orderline (OK)  --------------------------
-- check if there is any duplicate order
SELECT Count (sc_orderid
              || product_id)
FROM   monchef.school_canteen_orderline; -- 4346

SELECT Count (DISTINCT sc_orderid
                       || product_id)
FROM   monchef.school_canteen_orderline; -- 4346
-- check if there is ordered product that does not exist
SELECT Count (product_id)
FROM   monchef.school_canteen_orderline
WHERE  product_id NOT IN (SELECT product_id
                          FROM   product); -- 0
-- check if there is ordered product that does not exist
SELECT Count (sc_orderid)
FROM   monchef.school_canteen_orderline
WHERE  sc_orderid NOT IN (SELECT sc_orderid
                          FROM   school_canteen_order); -- 0
-- check if there is null value (all comlumn consist of not null constraint)
desc monchef.school_canteen_orderline;
-- check if there is any lineprice/quantitysold that doesnt make sense
SELECT *
FROM   monchef.school_canteen_orderline
WHERE  sol_quantitysold < 0
        OR sol_lineprice < 0;

CREATE TABLE school_canteen_orderline AS
  SELECT *
  FROM   monchef.school_canteen_orderline;

---------------------------------- catering_order (2 Errors) -------------------
SELECT *
FROM   monchef.catering_order;

-- check if there is any duplicate data
SELECT Count(*)
FROM   monchef.catering_order; -- 1504

SELECT Count (DISTINCT ca_orderid)
FROM   monchef.catering_order; -- 1500 duplicate exists
-- check if there is any null value
desc monchef.catering_order;
-- check if there is date that does not make sense
SELECT DISTINCT To_char(ca_orderdate, 'YYYY')
FROM   monchef.catering_order; -- there is year 2035 which does not make sense

SELECT DISTINCT To_char(ca_orderdate, 'MM') AS MM
FROM   monchef.catering_order
```

```
ORDER  BY mm;

SELECT DISTINCT To_char(ca_orderdate, 'DD') AS DD
FROM   monchef.catering_order
ORDER  BY dd;

-- check if there is any promo_code that does not exist
SELECT *
FROM   monchef.catering_order
WHERE  promo_code NOT IN (SELECT promo_code
                          FROM   promotion);

-- check if there is any provider_id that does not exist
SELECT *
FROM   monchef.catering_order
WHERE  provider_id NOT IN (SELECT provider_id
                           FROM   delivery_provider);

-- check if there is any staff_id that does not exist
SELECT *
FROM   monchef.catering_order
WHERE  staff_id NOT IN (SELECT staff_id
                        FROM   staff);

-- check if there is any status_id that does not exist
SELECT *
FROM   monchef.catering_order
WHERE  status_id NOT IN (SELECT status_id
                         FROM   order_status);

-- creating table
CREATE TABLE catering_order AS
  SELECT DISTINCT *
  FROM   monchef.catering_order
  WHERE  To_char(ca_orderdate, 'YYYY') != '2035';

---------------------------------- catering_orderline (1 Error) -----------------
-----------------
SELECT *
FROM   monchef.catering_orderline;

-- check if there is any duplicate order
SELECT Count (ca_orderid
              || product_id)
FROM   monchef.catering_orderline; -- 4495

SELECT Count (DISTINCT ca_orderid
                       || product_id)
FROM   monchef.catering_orderline; -- 4495
-- check if there is ordered product that does not exist
SELECT *
```

```
FROM   monchef.catering_orderline
WHERE  product_id NOT IN (SELECT product_id
                          FROM   product); -- CA_ORDERID: C1890
-- check if there is order that does not exist
SELECT *
FROM   monchef.catering_orderline
WHERE  ca_orderid NOT IN (SELECT ca_orderid
                          FROM   catering_order); -- CA_ORDERID: C1890
SELECT *
FROM   monchef.catering_orderline
WHERE  ca_orderid = 'C1890'; -- only 1 order is wrong
-- check if there is null value: null does not exist
desc monchef.catering_orderline;
-- check if there is any lineprice/quantitysold that doesnt make sense
SELECT *
FROM   monchef.catering_orderline
WHERE  col_quantitysold < 0
        OR col_lineprice < 0;

-- creating table
CREATE TABLE catering_orderline AS
  SELECT *
  FROM   monchef.catering_orderline
  WHERE  ca_orderid != 'C1890';
```

Data Warehouse

Star Schema Level 2



Technical Explanation

The Fact measures identified in the following star schema are:

- total_sales for Catering Services
- total_number_of_order for Catering Services
- total_delivery_cost for Catering Services
- total_sales for Canteen Service
- total_number_of_order for Canteen Service
- total_number_of_products
- total_number_of_customers

The Dimensions in Leve 2 are as follows:

- Suburb
- Customer Type
- Payment Type
- Time
- Season
- Promotion
- Gender
- Order Price Scale
- Order Status
- Delivery Provider
- Canteen Order
- Catering Order
- Product
- Category
- Product Price Scale

The Bridge table Dimensions are as follows:

- Canteen Order Line
- Catering Order Line
- Product Category Bridge

Determinant Dimensions:

- None
- Total sales are independent of cancelled order status. After further clarification from the client representative, it has been found that any cancelled ordered will not be refunded from the sales price, but from a different allocated fund pool.

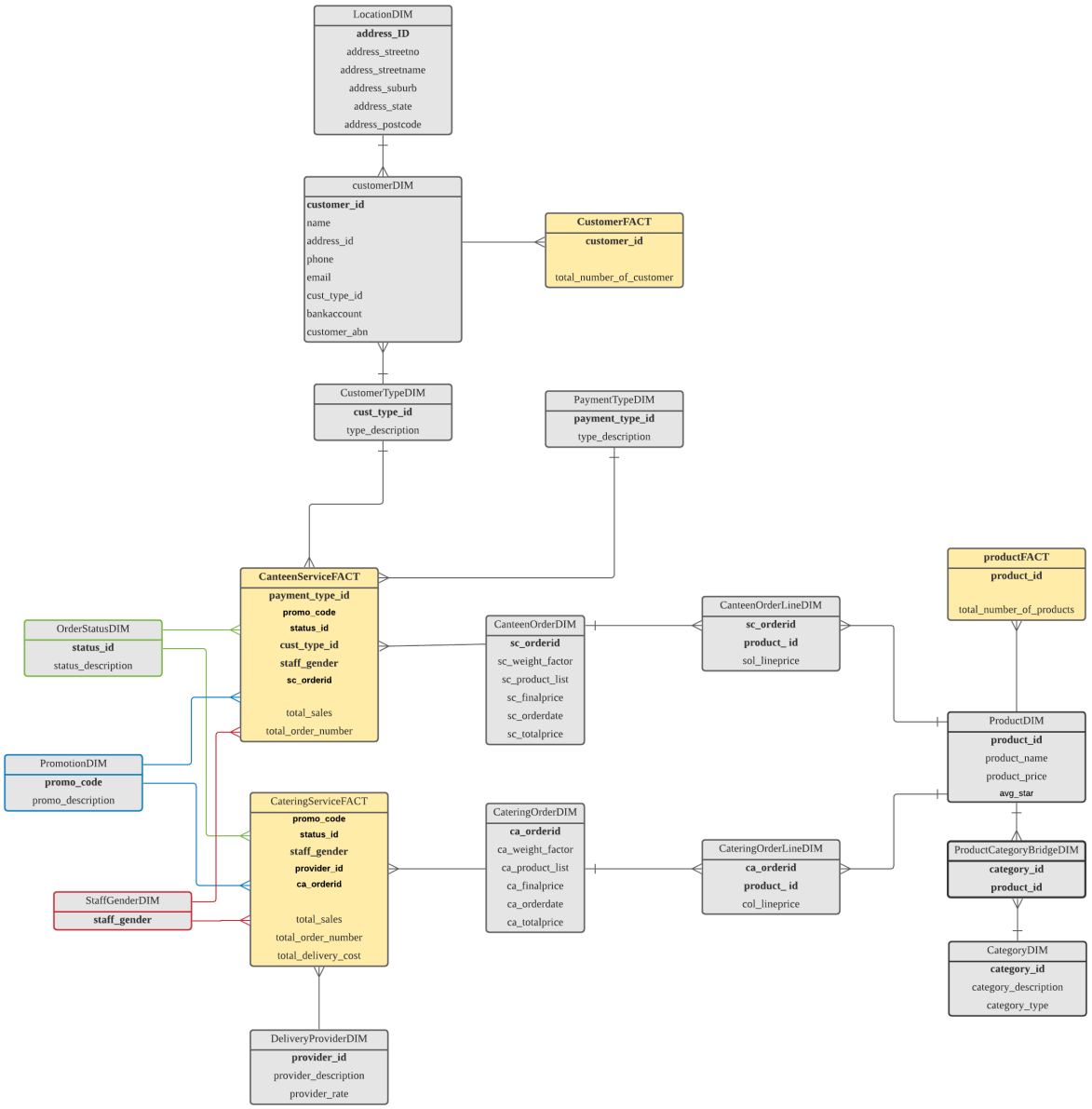
Slowly Changing Dimensions:

- None, after further clarification, the prices of the product do not change, or at least, there is no method to track changes in the given operational database

Hierarchical Structure:

Every Dimension is fully independent, and the star schema does not compose of any hierarchical structure to enable optimal and efficient implementation and reduce the complexity of generating queries and reports.

Star Schema Level 0



Technical Explanation

The Fact measures identified in the following star schema are (same as Level 2):

- total_sales for Catering Services
- total_number_of_order for Catering Services
- total_delivery_cost for Catering Services
- total_sales for Canteen Service
- total_number_of_order for Canteen Service
- total_number_of_products
- total_number_of_customers

The Dimensions in Leve 2 are as follows:

- Customer
  - Location
- Customer Type
- Payment Type
- Promotion
- Gender
- Order Status
- Delivery Provider
- Canteen Order
- Catering Order
- Product
- Category

Some dimensions were removed, but some other dimensions had been given more attributes in order to accommodate to level 0 star schema.
Time Dimension, Season, Product and Order Price Scale have been removed to reduce the aggregation level to 0. Additionally, the Suburb dimension has been changed to Location instead for higher granularity record and thus, lower aggregation.

The Bridge table Dimensions are as follows:

- Canteen Order Line
- Catering Order Line
- Product Category Bridge

Determinant Dimensions:

- None
- Total sales are independent of cancelled order status. After further clarification from the client representative, it has been found that any cancelled ordered will not be refunded from the sales price, but from a different allocated fund pool.

Slowly Changing Dimensions:

- None, after further clarification, the prices of the product do not change, or at least, there is no method to track changes in the given operational database

Hierarchical Structure:
To accommodate for the Level 0 star schema, Location Dimension is now part of a hierarchical structure under customer dimension, reducing the aggregation of the star schema without compromising the information needed for analysis purposes in respect to location.

Implementation of Star Schema Level 2

SQL Code: Drop Commands

```
DROP TABLE suburbdim;
DROP TABLE customertypedim;
DROP TABLE paymenttypedim;
DROP TABLE promotiondim;
DROP TABLE orderstatusdim;
DROP TABLE genderdim;
DROP TABLE deliveryproviderdim;
DROP TABLE productpricescaledim;
DROP TABLE orderpricescaledim;
DROP TABLE temptimedim;
DROP TABLE timedim;
DROP TABLE tempseasondim;
DROP TABLE seasondim;
DROP TABLE productcategorybridgedim;
DROP TABLE categorydim;
DROP TABLE tempproductdim;
DROP TABLE productdim;
DROP TABLE canteenorderdim;
DROP TABLE canteenorderlinedim;
DROP TABLE cateringorderdim;
DROP TABLE cateringorderlinedim;

--fact tables:
DROP TABLE customerfact;
DROP TABLE tempcanteenservicefact;
DROP TABLE canteenservicefact;

DROP TABLE tempcateringservicefact;
DROP TABLE cateringservicefact;
DROP TABLE tempproductfact;
DROP TABLE productfact;
```

SQL Code: Dimensions

```
/*-----------------------------------------------
Dimension 1: SuburbDIM
-----------------------------------------------*/
CREATE TABLE suburbdim AS
  SELECT DISTINCT address_suburb
  FROM   address;
```

	ADDRESS_SUBURB
1	Ripponlea
2	Middle Park
3	Southbank
4	St Kilda
5	South Melbourne
6	Elwood
7	Melbourne
8	Balaclava

```
/*-----------------------------------------------
Dimension 2: CustomerTypeDIM
-----------------------------------------------*/
CREATE TABLE customertypedim AS
  SELECT type_id AS cust_type_id,
         type_description
  FROM   customer_type;
```

	CUST_TYPE_ID	TYPE_DESCRIPTION
1	CT001	Individual
2	CT002	Business
3	CT003	School

```
/*-----------------------------------------------
Dimension 3: PaymentTypeDIM
-----------------------------------------------*/
CREATE TABLE paymenttypedim AS
  SELECT type_id AS payment_type_id,
         type_description
  FROM   payment_type;
```

	PAYMENT_TYPE_ID	TYPE_DESCRIPTION
1	PT001	Cash
2	PT002	PayPal
3	PT003	Credit Card
4	PT004	Bank Transfer

```
/*-----------------------------------------------
Dimension 4: PromotionDIM
-----------------------------------------------*/
CREATE TABLE promotiondim AS
  SELECT *
  FROM   promotion;

INSERT INTO promotiondim
VALUES      ('no_pr',
             'no promotion used');
```

	PROMO_CODE	PROMO_DESCRIPTION
1	PR001	5% OFF
2	PR002	10% OFF
3	PR003	15% OFF
4	no_pr	no promotion used

```
/*-----------------------------------------------
Dimension 5: orderStatusDIM
-----------------------------------------------*/
CREATE TABLE orderstatusdim AS
  SELECT *
  FROM   order_status;
```

	STATUS_ID	STATUS_DESCRIPTION
1	OS001	Pending
2	OS002	Accepted
3	OS003	Processed
4	OS004	Delivered
5	OS005	Completed
6	OS006	Cancelled

```
/*-----------------------------------------------
Dimension 6: GenderDIM
-----------------------------------------------*/
CREATE TABLE genderdim AS
  SELECT DISTINCT staff_gender
  FROM   staff;

ALTER TABLE genderdim
  ADD ( description VARCHAR2(6));

UPDATE genderdim
SET    description = 'Female'
WHERE  staff_gender = 'F';

UPDATE genderdim
SET    description = 'Male'
WHERE  staff_gender = 'M';
```

	STAFF_GENDER	DESCRIPTION
1	M	Male
2	F	Female

```
/*-----------------------------------------------
Dimension 7: DeliveryProviderDIM
-----------------------------------------------*/
CREATE TABLE deliveryproviderdim AS
  SELECT *
  FROM   delivery_provider;
```

	PROVIDER_ID	PROVIDER_DESCRIPTION	PROVIDER_RATE
1	DP001	Instore pick-up	0
2	DP002	UberEAT	0.15
3	DP003	DoorDash	0.1
4	DP004	Deliveroo	0.12
5	DP005	Menulog	0.1

```
/*-----------------------------------------------
Dimension 8: productPriceScaleDIM
-----------------------------------------------*/
CREATE TABLE productpricescaledim
  (
     product_price_scaleid CHAR(1),
     description           VARCHAR2(50),
     minprice              NUMBER(5, 2),
     maxprice              NUMBER(5, 2)
  );

INSERT INTO productpricescaledim
VALUES      ('l',
             'low price range',
             0.00,
             9.99);

INSERT INTO productpricescaledim
```

```
VALUES      ('m',
             'medium price range',
             10.00,
             20.00);

INSERT INTO productpricescaledim
VALUES      ('h',
             'high price range',
             20.01,
             NULL);
```

	PRODUCT_PRICE_SCALEID	DESCRIPTION	MINPRICE	MAXPRICE
1	l	low price range	0	9.99
2	m	medium price range	10	20
3	h	high price range	20.01	(null)

```
/*-------------------------------------------------
Dimension 9: orderPriceScaleDIM
-------------------------------------------------*/
CREATE TABLE orderpricescaledim
  (
     order_price_scaleid CHAR(1),
     description         VARCHAR2(50),
     minprice            NUMBER(6, 2),
     maxprice            NUMBER(6, 2)
  );

INSERT INTO orderpricescaledim
VALUES      ('l',
             'low price range',
             0.00,
             49.99);

INSERT INTO orderpricescaledim
VALUES      ('m',
             'medium price range',
             50.00,
             150.00);

INSERT INTO orderpricescaledim
VALUES      ('e',
             'expensive price range',
             150.01,
             NULL);
```

	ORDER_PRICE_SCALEID	DESCRIPTION	MINPRICE	MAXPRICE
1	l	low price range	0	49.99
2	m	medium price range	50	150
3	e	expensive price range	150.01	(null)

```
/*-----------------------------------------------
Dimension 10: TimeDIM
-----------------------------------------------*/
-- first: create the temporary dimension
CREATE TABLE temptimedim AS
  SELECT DISTINCT To_char(sc_orderdate, 'Day')  AS weeklyDay,
                  To_char(sc_orderdate, 'DD')   AS the_day,
                  To_char(sc_orderdate, 'MM')   AS the_month,
                  To_char(sc_orderdate, 'YYYY') AS the_year
  FROM   school_canteen_order
  UNION
  SELECT DISTINCT To_char(ca_orderdate, 'Day')  AS weeklyDay,
                  To_char(ca_orderdate, 'DD')   AS the_day,
                  To_char(ca_orderdate, 'MM')   AS the_month,
                  To_char(ca_orderdate, 'YYYY') AS the_year
  FROM   catering_order;

-- adding timeID
ALTER TABLE temptimedim
  ADD ( timeid CHAR(8));

-- filling the time id in the following format: YYYYMMDD
UPDATE temptimedim
SET    timeid = the_year
                ||the_month
                ||the_day;

-- Final Dimension Table: TimeDIM
CREATE TABLE timedim AS
  SELECT timeid,
         weeklyday,
         the_day,
         the_month,
         the_year
  FROM   temptimedim;
```

	WEEKLYDAY	THE_DAY	THE_MONTH	THE_YEAR	TIMEID
1	Friday	01	02	2019	20190201
2	Friday	01	03	2019	20190301
3	Friday	01	05	2020	20200501
4	Friday	01	11	2019	20191101
5	Friday	02	03	2018	20180302

Total rows: 743

	TIMEID	WEEKLYDAY	THE_DAY	THE_MONTH	THE_YEAR
1	20190201	Friday	01	02	2019
2	20190301	Friday	01	03	2019
3	20200501	Friday	01	05	2020
4	20191101	Friday	01	11	2019
5	20180302	Friday	02	03	2018

Total rows: 743

```
/*-----------------------------------------------
Dimension 11: Season
-----------------------------------------------*/
-- temp dimension
CREATE TABLE tempseasondim AS
  SELECT DISTINCT To_char(sc_orderdate, 'MM') AS the_month
  FROM   school_canteen_order
  UNION
  SELECT DISTINCT To_char(ca_orderdate, 'MM') AS the_month
  FROM   catering_order;

ALTER TABLE tempseasondim
  ADD ( season VARCHAR2(6));

UPDATE tempseasondim
SET    season = 'summer'
WHERE  the_month = '12'
        OR the_month <= '02';

UPDATE tempseasondim
SET    season = 'winter'
WHERE  the_month >= '06'
       AND the_month <= '08';

UPDATE tempseasondim
SET    season = 'autumn'
WHERE  the_month >= '03'
       AND the_month <= '05';

UPDATE tempseasondim
SET    season = 'spring'
WHERE  the_month >= '09'
       AND the_month <= '11';

-- Final Dimension:
CREATE TABLE seasondim AS
  SELECT DISTINCT season
  FROM   tempseasondim;
```

	THE_MONTH	SEASON
1	01	summer
2	02	summer
3	03	autumn
4	04	autumn
5	05	autumn
6	06	winter
7	07	winter
8	08	winter
9	09	spring
10	10	spring
11	11	spring
12	12	summer

	SEASON
1	winter
2	summer
3	spring
4	autumn

```
/*-----------------------------------------------
Dimension 12: ProductDIM
-----------------------------------------------*/
CREATE TABLE tempproductdim AS
  SELECT p.product_id,
         p.product_name,
         p.product_price,
         Nvl(r.review_star, 0) AS Star
  FROM   product p,
         product_review r
  WHERE  p.product_id = r.product_id(+);

CREATE TABLE productdim AS
  SELECT product_id,
         product_name,
         product_price,
         Round(Avg(star)) AS Avg_Star
  FROM   tempproductdim
  GROUP  BY product_id,
            product_name,
            product_price;
```

	PRODUCT_ID	PRODUCT_NAME	PRODUCT_PRICE	STAR
1	F0064	Green Curry Vegetables (Mhuyang)	20.9	5
2	F0046	Pad Gra Pao Seafood	21.9	5
3	F0080	Pineapple Breeze	7.9	5
4	F0150	Cheesy Fire Chicken	28.9	5
5	F0077	Chips	7.9	5

Total rows: 1605

	PRODUCT_ID	PRODUCT_NAME	PRODUCT_PRICE	AVG_STAR
1	F0099	Sprite	4.9	5
2	F0131	BBQ Marinated Beef Ribs Fingers	24.9	5
3	F0067	Mhu Yang Som Tum	20.9	5
4	F0120	Deep Fried Squid	13.9	5
5	F0114	Topokki	14.9	5

Total rows: 190

```
/*-----------------------------------------------
BRIDGE DIMENSION TABLE: ProductCategoryBridgeDIM
-----------------------------------------------*/
-- temp dimension: tempProductwithStar
CREATE TABLE productcategorybridgedim AS
  SELECT *
  FROM   product_category;
```

	PRODUCT_ID	CATEGORY_ID
1	F0001	FC002
2	F0001	FC004
3	F0001	FC009
4	F0002	FC002
5	F0002	FC004

Total rows: 547

```
/*-----------------------------------------------
Dimension 13: CategoryDIM
-----------------------------------------------*/
CREATE TABLE categorydim AS
  SELECT *
  FROM   category;

ALTER TABLE categorydim
  ADD ( category_type VARCHAR2(10));

-- setting category type to Meal
UPDATE categorydim
SET    category_type = 'meal'
WHERE  category_description IN ( 'Main', 'Starter', 'Dessert', 'Drink' );

UPDATE categorydim
SET    category_type = 'flavour'
WHERE  category_description IN ( 'Sweet', 'Savoury' );

UPDATE categorydim
SET    category_type = 'cuisine'
WHERE  category_description IN ( 'Indonesian', 'Korean', 'Thai' );
```

	CATEGORY_ID	CATEGORY_DESCRIPTION	CATEGORY_TYPE
1	FC001	Savoury	flavour
2	FC002	Sweet	flavour
3	FC003	Starter	meal
4	FC004	Main	meal
5	FC005	Dessert	meal
6	FC006	Drink	meal
7	FC007	Indonesian	cuisine
8	FC008	Korean	cuisine
9	FC009	Thai	cuisine

```
/*-----------------------------------------------
Dimension 14: CanteenOrderDIM
-----------------------------------------------*/
CREATE TABLE canteenorderdim AS
  SELECT o.sc_orderid,
         o.sc_totalprice,
         1.0 / Count(ol.product_id)              AS sc_weight_factor,
         Listagg (ol.product_id, '_')
           within GROUP (ORDER BY ol.product_id) AS sc_product_list
  FROM   school_canteen_order o,
         school_canteen_orderline ol
  WHERE  o.sc_orderid = ol.sc_orderid
  GROUP  BY o.sc_orderid,
            sc_totalprice;
```

	SC_ORDERID	SC_TOTALPRICE	SC_WEIGHT_FACTOR	SC_PRODUCT_LIST
1	S0001	184.5	1	F0110
2	S0002	248.8	0.25	F0022_F0130_F0148_F0167
3	S0003	245.6	0.333333333333333333333333333333333333333333	F0055_F0081_F0134
4	S0004	249.1	0.25	F0064_F0109_F0110_F0156
5	S0005	112.5	0.5	F0090_F0183

Total rows: 1500

```
/*-----------------------------------------------
Dimension 15: CanteenOrderLineDIM
-----------------------------------------------*/
CREATE TABLE canteenorderlinedim AS
  SELECT sc_orderid,
         product_id,
         sol_lineprice
  FROM   school_canteen_orderline;
```

	SC_ORDERID	PRODUCT_ID	SOL_LINEPRICE
1	S0204	F0052	41.8
2	S0204	F0049	18.9
3	S0204	F0072	22.5
4	S0204	F0031	31.8
5	S0205	F0044	109.5

Total rows: 4346

```
CREATE TABLE cateringorderdim AS
  SELECT o.ca_orderid,
         o.ca_totalprice,
         1.0 / Count(ol.product_id)              AS ca_weight_factor,
         Listagg (ol.product_id, '_')
           within GROUP (ORDER BY ol.product_id) AS ca_product_list
  FROM   catering_order o,
         catering_orderline ol
  WHERE  o.ca_orderid = ol.ca_orderid
  GROUP  BY o.ca_orderid,
            o.ca_totalprice;
```

	CA_ORDERID	CA_TOTALPRICE	CA_WEIGHT_FACTOR	CA_PRODUCT_LIST
1	C0001	7.9	1	F0081
2	C0002	60	1	F0183
3	C0003	174.9	0.2	F0029_F0060_F0076_F0096_F0182
4	C0004	110.7	1	F0113
5	C0005	74.4	0.25	F0010_F0099_F0162_F0163

Total rows: 1500

```
/*-----------------------------------------------
Dimension 17: CateringOrderLineDIM
-----------------------------------------------*/
CREATE TABLE cateringorderlinedim AS
  SELECT ca_orderid,
         product_id,
         col_lineprice
  FROM   catering_orderline;
```

	CA_ORDERID	PRODUCT_ID	COL_LINEPRICE
1	C0200	F0081	15.8
2	C0200	F0083	39.5
3	C0200	F0064	62.7
4	C0200	F0091	13
5	C0201	F0177	520

Total rows:4494

SQL Code: Fact Tables

```
/*--------------------------------------------------
CustomerFact
--------------------------------------------------*/
CREATE TABLE customerfact AS
  SELECT a.address_suburb AS suburd,
         type_id          AS cust_type_id,
         Count(*)         AS total_number_of_customer
  FROM   customer c
         join address a
           ON c.address_id = a.address_id
  GROUP  BY a.address_suburb,
            type_id;
```

	SUBURD	CUST_TYPE_ID	TOTAL_NUMBER_OF_CUSTOMER
1	Balaclava	CT002	1
2	St Kilda	CT003	4
3	Middle Park	CT003	2
4	Southbank	CT003	4
5	Melbourne	CT002	1
6	Melbourne	CT001	2
7	Elwood	CT003	1
8	South Melbourne	CT001	1
9	Southbank	CT001	2
10	Ripponlea	CT001	4
11	Elwood	CT002	4
12	Ripponlea	CT002	2
13	Elwood	CT001	2
14	St Kilda	CT002	3
15	South Melbourne	CT002	1
16	Balaclava	CT001	1
17	Southbank	CT002	2
18	South Melbourne	CT003	4
19	Middle Park	CT001	4
20	Melbourne	CT003	3
21	Middle Park	CT002	2

```
/*-----------------------------------------------
CanteenServiceFACT
-----------------------------------------------*/
CREATE TABLE tempcanteenservicefact AS
  SELECT sc_orderid,
         To_char(sc_orderdate, 'YYYY')
         || To_char(sc_orderdate, 'MM')
         || To_char(sc_orderdate, 'DD') AS timeID,
         To_char(sc_orderdate, 'MM')    AS order_month,
         sc.type_id                     AS payment_type_id,
         sc.promo_code,
         c.type_id                      AS cust_type_id,
         s.staff_gender,
         sc.status_id,
         sc.sc_finalprice
  FROM   school_canteen_order sc
         join customer c
           ON sc.customer_id = c.customer_id
         join staff s
           ON sc.staff_id = s.staff_id;

ALTER TABLE tempcanteenservicefact
  ADD ( season VARCHAR2(10));

UPDATE tempcanteenservicefact
SET    season = 'summer'
WHERE  order_month = '12'
        OR order_month <= '02';

UPDATE tempcanteenservicefact
SET    season = 'winter'
WHERE  order_month >= '06'
       AND order_month <= '08';

UPDATE tempcanteenservicefact
SET    season = 'autumn'
WHERE  order_month >= '03'
       AND order_month <= '05';

UPDATE tempcanteenservicefact
SET    season = 'spring'
WHERE  order_month >= '09'
       AND order_month <= '11';

ALTER TABLE tempcanteenservicefact
  ADD ( order_price_scaleid VARCHAR2(10));

UPDATE tempcanteenservicefact
SET    order_price_scaleid = 'l'
WHERE  sc_finalprice >= 0.00
```

```
        OR sc_finalprice <= 49.99;

UPDATE tempcanteenservicefact
SET    order_price_scaleid = 'm'
WHERE  sc_finalprice >= 50.00
       AND sc_finalprice <= 150.00;

UPDATE tempcanteenservicefact
SET    order_price_scaleid = 'e'
WHERE  sc_finalprice >= 150.01;

CREATE TABLE canteenservicefact AS
  SELECT sc_orderid,
         timeid,
         season,
         order_price_scaleid,
         payment_type_id,
         Nvl(promo_code, 'no_pr') AS promo_code,
         status_id,
         staff_gender,
         cust_type_id,
         SUM(sc_finalprice)       AS total_sales,
         Count(*)                 AS total_order_number
  FROM   tempcanteenservicefact
  GROUP  BY sc_orderid,
            timeid,
            season,
            order_price_scaleid,
            payment_type_id,
            promo_code,
            status_id,
            staff_gender,
            cust_type_id;
```

	SC_ORDERID	TIMEID	ORDER_MONTH	PAYMENT_TYPE_ID	PROMO_CODE	CUST_TYPE_ID	STAFF_GENDER	STATUS_ID	SC_FINALPRICE	SEASON	ORDER_PRICE_SCALEID
1	S0096	20190416	04	PT001	(null)	CT003	M	OS005	275.9	autumn	e
2	S0525	20190416	04	PT004	PR001	CT001	F	OS005	25.01	autumn	l
3	S1427	20190416	04	PT002	PR001	CT003	F	OS005	97.38	autumn	m
4	S1467	20190416	04	PT001	PR002	CT001	F	OS005	126.02	autumn	m
5	S0138	20190417	04	PT003	PR003	CT003	M	OS006	182.32	autumn	e

Temp Fact Total rows: 1500

	SC_ORDERID	TIMEID	SEASON	ORDER_PRICE_SCALEID	PAYMENT_TYPE_ID	PROMO_CODE	STATUS_ID	STAFF_GENDER	CUST_TYPE_ID	TOTAL_SALES	TOTAL_ORDER_NUMBER
1	S0909	20190512	autumn	e	PT002	PR001	OS005	F	CT001	370.5	1
2	S1393	20190513	autumn	l	PT004	PR002	OS005	F	CT002	29.7	1
3	S1283	20190519	autumn	m	PT002	PR003	OS005	M	CT002	112.2	1
4	S0029	20190520	autumn	m	PT002	PR002	OS006	M	CT002	67.7	1
5	S0855	20190521	autumn	e	PT002	PR003	OS005	F	CT001	255.93	1

Final Fact Total rows: 1500

```
/*------------------------------------------------------
CateringServiceFACT
------------------------------------------------------*/
CREATE TABLE tempcateringservicefact AS
  SELECT ca_orderid,
         To_char(ca_orderdate, 'YYYY')
         || To_char(ca_orderdate, 'MM')
         || To_char(ca_orderdate, 'DD') AS timeID,
         To_char(ca_orderdate, 'MM')    AS order_month,
         ca.promo_code,
         s.staff_gender,
         ca.status_id,
         ca.ca_finalprice,
         ca.provider_id,
         ca.ca_totalprice,
         d.provider_rate
  FROM   catering_order ca
         join staff s
           ON ca.staff_id = s.staff_id
         join delivery_provider d
           ON d.provider_id = ca.provider_id;

ALTER TABLE tempcateringservicefact
  ADD ( season VARCHAR2(10));

UPDATE tempcateringservicefact
SET    season = 'summer'
WHERE  order_month = '12'
        OR order_month <= '02';

UPDATE tempcateringservicefact
SET    season = 'winter'
WHERE  order_month >= '06'
       AND order_month <= '08';

UPDATE tempcateringservicefact
SET    season = 'autumn'
WHERE  order_month >= '03'
       AND order_month <= '05';

UPDATE tempcateringservicefact
SET    season = 'spring'
WHERE  order_month >= '09'
       AND order_month <= '11';

ALTER TABLE tempcateringservicefact
  ADD ( order_price_scaleid VARCHAR2(10));

UPDATE tempcateringservicefact
SET    order_price_scaleid = 'l'
WHERE  ca_finalprice >= 0.00
```

```
        OR ca_finalprice <= 49.99;

UPDATE tempcateringservicefact
SET    order_price_scaleid = 'm'
WHERE  ca_finalprice >= 50.00
       AND ca_finalprice <= 150.00;

UPDATE tempcateringservicefact
SET    order_price_scaleid = 'e'
WHERE  ca_finalprice >= 150.01;

CREATE TABLE cateringservicefact AS
  SELECT ca_orderid,
         timeid,
         season,
         order_price_scaleid,
         Nvl(promo_code, 'no_pr')           AS promo_code,
         status_id,
         staff_gender,
         provider_id,
         SUM(ca_finalprice)                 AS total_sales,
         Count(*)                           AS total_order_number,
         SUM(ca_totalprice * provider_rate) AS total_delivery_cost
  FROM   tempcateringservicefact
  GROUP  BY ca_orderid,
            timeid,
            season,
            order_price_scaleid,
            promo_code,
            status_id,
            staff_gender,
            provider_id;
```

	CA_ORDERID	TIMEID	ORDER_MONTH	PROMO_CODE	STAFF_GENDER	STATUS_ID	CA_FINALPRICE	PROVIDER_ID	CA_TOTALPRICE	PROVIDER_RATE	SEASON	ORDER_PRICE_SCALEID
1	C0324	20180811	08	(null)	M	OS005	33.6	DP004	30	0.12	winter	l
2	C1229	20180819	08	PR003	F	OS005	13.09	DP004	13.5	0.12	winter	l
3	C0614	20180822	08	(null)	M	OS005	201.82	DP002	175.5	0.15	winter	e
4	C1475	20180905	09	PR003	M	OS005	51.85	DP001	61	0	spring	m
5	C0883	20180907	09	PR003	M	OS005	7.65	DP001	9	0	spring	l

Temp Fact Total rows: 1500

	CA_ORDERID	TIMEID	SEASON	ORDER_PRICE_SCALEID	PROMO_CODE	STATUS_ID	STAFF_GENDER	PROVIDER_ID	TOTAL_SALES	TOTAL_ORDER_NUMBER	TOTAL_DELIVERY_COST
1	C1037	20181008	spring	m	PR001	OS005	M	DP003	108.89	1	10.37
2	C0355	20181015	spring	e	no_pr	OS005	F	DP004	206.64	1	22.14
3	C0193	20181201	summer	e	no_pr	OS005	F	DP003	185.68	1	16.88
4	C1283	20181213	summer	e	PR003	OS005	F	DP004	256.08	1	31.68
5	C1431	20190111	summer	l	PR002	OS005	M	DP001	9	1	0

Final FactTotal rows: 1500

```
/*------------------------------------------------
productFACT
--------------------------------------------------*/
CREATE TABLE tempproductfact AS
  SELECT product_id,
         product_price
  FROM   product;

ALTER TABLE tempproductfact
  ADD ( product_price_scaleid VARCHAR2(10));

UPDATE tempproductfact
SET    product_price_scaleid = 'l'
WHERE  product_price >= 0.00
       AND product_price <= 9.99;

UPDATE tempproductfact
SET    product_price_scaleid = 'm'
WHERE  product_price >= 10.00
       AND product_price <= 20.00;

UPDATE tempproductfact
SET    product_price_scaleid = 'h'
WHERE  product_price >= 20.01;

CREATE TABLE productfact AS
  SELECT product_id,
         product_price_scaleid,
         Count(*) AS total_number_of_products
  FROM   tempproductfact
  GROUP  BY product_id,
            product_price_scaleid;
```

	PRODUCT_ID	PRODUCT_PRICE	PRODUCT_PRICE_SCALEID
1	F0001	16.9	m
2	F0002	16.9	m
3	F0003	10	m
4	F0004	16.9	m
5	F0005	5.5	l

Temp Fact Total rows: 190

	PRODUCT_ID	PRODUCT_PRICE_SCALEID	TOTAL_NUMBER_OF_PRODUCTS
1	F0003	m	1
2	F0013	m	1
3	F0024	m	1
4	F0032	m	1
5	F0033	m	1

Final Fact Total rows: 190

Implementation of Star Schema Level 0

SQL Code: Drop Commands

```
--dimensions
DROP TABLE locationdim_0;
DROP TABLE customertypedim_0;
DROP TABLE paymenttypedim_0;
DROP TABLE promotiondim_0;
DROP TABLE orderstatusdim_0;
DROP TABLE customerdim_0;
DROP TABLE genderdim_0;
DROP TABLE deliveryproviderdim_0;
DROP TABLE tempproductdim_0;
DROP TABLE productdim_0;
DROP TABLE productcategorybridgedim_0;
DROP TABLE categorydim_0;
DROP TABLE canteenorderdim_0;
DROP TABLE canteenorderlinedim_0;
DROP TABLE cateringorderdim_0;
DROP TABLE cateringorderlinedim_0;

--fact tables:
DROP TABLE canteenservicefact_0;
DROP TABLE customerfact_0;
DROP TABLE cateringservicefact_0;
DROP TABLE productfact_0;
```

SQL Code: Dimensions

```
/*-----------------------------------------------
Dimension 1: LocationDIM
-----------------------------------------------*/
CREATE TABLE locationdim_0 AS
  SELECT *
  FROM   address;
```

	ADDRESS_ID	ADDRESS_STREETNO	ADDRESS_STREETNAME	ADDRESS_SUBURB	ADDRESS_STATE	ADDRESS_POSTCODE
1	AD001	26	Walker Street	Elwood	VIC	3184
2	AD002	723	Parsons Street	Southbank	VIC	3006
3	AD003	229	Barnett Street	Ripponlea	VIC	3185
4	AD004	60	Magistrates Walk	Elwood	VIC	3184
5	AD005	565	Witchwood Close	St Kilda	VIC	3182

Total rows: 249

```
/*------------------------------------------------
Dimension 2: CustomerTypeDIM
------------------------------------------------*/
CREATE TABLE customertypedim_0 AS
  SELECT type_id AS cust_type_id,
         type_description
  FROM   customer_type;
```

	CUST_TYPE_ID	TYPE_DESCRIPTION
1	CT001	Individual
2	CT002	Business
3	CT003	School

```
/*------------------------------------------------
Dimension 3: PaymentTypeDIM
------------------------------------------------*/
CREATE TABLE paymenttypedim_0 AS
  SELECT type_id AS payment_type_id,
         type_description
  FROM   payment_type;
```

	PAYMENT_TYPE_ID	TYPE_DESCRIPTION
1	PT001	Cash
2	PT002	PayPal
3	PT003	Credit Card
4	PT004	Bank Transfer

```
/*------------------------------------------------
Dimension 4: PromotionDIM
------------------------------------------------*/
CREATE TABLE promotiondim_0 AS
  SELECT *
  FROM   promotion;

INSERT INTO promotiondim_0
VALUES      ('no_pr',
             'no promotion used');
```

	PROMO_CODE	PROMO_DESCRIPTION
1	PR001	5% OFF
2	PR002	10% OFF
3	PR003	15% OFF
4	no_pr	no promotion used

```
/*------------------------------------------------
Dimension 5: orderStatusDIM
------------------------------------------------*/
CREATE TABLE orderstatusdim_0 AS
  SELECT *
  FROM   order_status;
```

	STATUS_ID	STATUS_DESCRIPTION
1	OS001	Pending
2	OS002	Accepted
3	OS003	Processed
4	OS004	Delivered
5	OS005	Completed
6	OS006	Cancelled

```
/*-------------------------------------------------
Dimension 6: GenderDIM
-------------------------------------------------*/
CREATE TABLE genderdim_0 AS
  SELECT DISTINCT staff_gender
  FROM   staff;

ALTER TABLE genderdim_0
  ADD ( description VARCHAR2(6));

UPDATE genderdim_0
SET    description = 'Female'
WHERE  staff_gender = 'F';

UPDATE genderdim_0
SET    description = 'Male'
WHERE  staff_gender = 'M';
```

	STAFF_GENDER	DESCRIPTION
1	M	Male
2	F	Female

```
/*-------------------------------------------------
Dimension 7: DeliveryProviderDIM
-------------------------------------------------*/
CREATE TABLE deliveryproviderdim_0 AS
  SELECT *
  FROM   delivery_provider;
```

	PROVIDER_ID	PROVIDER_DESCRIPTION	PROVIDER_RATE
1	DP001	Instore pick-up	0
2	DP002	UberEAT	0.15
3	DP003	DoorDash	0.1
4	DP004	Deliveroo	0.12
5	DP005	Menulog	0.1

```
/*-------------------------------------------------
Dimension 8: CustomerDIM
-------------------------------------------------*/
CREATE TABLE customerdim_0 AS
  SELECT customer_id,
         customer_name,
         address_id,
         customer_phone,
         customer_email,
         type_id AS cust_type_id,
         customer_bankaccount,
         customer_abn
  FROM   customer;
```

	CUSTOMER_ID	CUSTOMER_NAME	ADDRESS_ID	CUSTOMER_PHONE	CUSTOMER_EMAIL	CUST_TYPE_ID	CUSTOMER_BANKACCOUNT	CUSTOMER_ABN
1	CU005	Alicia Goldstein	AD005	0488779937	alicia.goldstein@gmail.com	CT002	034532-55428775	78614901624
2	CU008	Jai Kater	AD008	0436209539	jai.kater@gmail.com	CT001	014556-77064742	null
3	CU009	Bella Petersen	AD009	0439306396	bella.petersen@gmail.com	CT003	034929-96133172	659636922221
4	CU012	David Darcy	AD012	0489217606	david.darcy@gmail.com	CT001	064539-85461030	null
5	CU020	Jonathan Clamp	AD020	0488797274	jonathan.clamp@gmail.com	CT001	014556-17146442	null

Total rows:50

```
/*-----------------------------------------------
Dimension 9: ProductDIM
-----------------------------------------------*/
CREATE TABLE tempproductdim_0 AS
  SELECT p.product_id,
         p.product_name,
         p.product_price,
         Nvl(r.review_star, 0) AS Star
  FROM   product p,
         product_review r
  WHERE  p.product_id = r.product_id(+);

CREATE TABLE productdim_0 AS
  SELECT product_id,
         product_name,
         product_price,
         Round(Avg(star)) AS Avg_Star
  FROM   tempproductdim_0
  GROUP  BY product_id,
            product_name,
            product_price;
```

	PRODUCT_ID	PRODUCT_NAME	PRODUCT_PRICE	STAR
1	F0064	Green Curry Vegetables (Mhuyang)	20.9	5
2	F0046	Pad Gra Pao Seafood	21.9	5
3	F0080	Pineapple Breeze	7.9	5
4	F0150	Cheesy Fire Chicken	28.9	5
5	F0077	Chips	7.9	5

Temp Dimension Total rows: 1605

	PRODUCT_ID	PRODUCT_NAME	PRODUCT_PRICE	AVG_STAR
1	F0099	Sprite	4.9	5
2	F0131	BBQ Marinated Beef Ribs Fingers	24.9	5
3	F0067	Mhu Yang Som Tum	20.9	5
4	F0120	Deep Fried Squid	13.9	5
5	F0114	Topokki	14.9	5

Final Dimension Total rows: 190

```
/*-----------------------------------------------
BRIDGE DIMENSION TABLE: ProductCategoryBridgeDIM
-----------------------------------------------*/
-- temp dimension: tempProductwithStar
CREATE TABLE productcategorybridgedim_0 AS
  SELECT *
  FROM   product_category;
```

	PRODUCT_ID	CATEGORY_ID
1	F0001	FC002
2	F0001	FC004
3	F0001	FC009
4	F0002	FC002
5	F0002	FC004

Total rows: 547

```
/*-----------------------------------------------
Dimension 10: CategoryDIM
-----------------------------------------------*/
CREATE TABLE categorydim_0 AS
  SELECT *
  FROM   category;

ALTER TABLE categorydim_0
  ADD ( category_type VARCHAR2(10));

-- setting category type to Meal
UPDATE categorydim_0
SET    category_type = 'meal'
WHERE  category_description IN ( 'Main', 'Starter', 'Dessert', 'Drink' );

UPDATE categorydim_0
SET    category_type = 'flavour'
WHERE  category_description IN ( 'Sweet', 'Savoury' );

UPDATE categorydim_0
SET    category_type = 'cuisine'
WHERE  category_description IN ( 'Indonesian', 'Korean', 'Thai' );
```

	CATEGORY_ID	CATEGORY_DESCRIPTION	CATEGORY_TYPE
1	FC001	Savoury	flavour
2	FC002	Sweet	flavour
3	FC003	Starter	meal
4	FC004	Main	meal
5	FC005	Dessert	meal
6	FC006	Drink	meal
7	FC007	Indonesian	cuisine
8	FC008	Korean	cuisine
9	FC009	Thai	cuisine

```
/*------------------------------------------------
Dimension 11: CanteenOrderDIM
------------------------------------------------*/
CREATE TABLE canteenorderdim_0 AS
  SELECT o.sc_orderid,
         o.sc_totalprice,
         o.sc_finalprice,
         o.sc_orderdate,
         1.0 / Count(ol.product_id)              AS sc_weight_factor,
         Listagg (ol.product_id, '_')
           within GROUP (ORDER BY ol.product_id) AS sc_product_list
  FROM   school_canteen_order o,
         school_canteen_orderline ol
  WHERE  o.sc_orderid = ol.sc_orderid
  GROUP  BY o.sc_orderid,
            o.sc_totalprice,
            o.sc_finalprice,
            o.sc_orderdate;
            sc_totalprice;
```

	SC_ORDERID	SC_TOTALPRICE	SC_FINALPRICE	SC_ORDERDATE	SC_WEIGHT_FACTOR	SC_PRODUCT_LIST
1	S0001	184.5	166.05	15/05/2019	1	F0110
2	S0002	248.8	243.92	02/08/2020	0.25	F0022_F0130_F0148_F0167
3	S0003	245.6	245.6	01/10/2019	0.3333333333333333333333333333333333333333	F0055_F0081_F0134
4	S0004	249.1	246.65	16/07/2019	0.25	F0064_F0109_F0110_F0156
5	S0005	112.5	101.25	16/07/2020	0.5	F0090_F0183

Total rows: 1500

```
/*------------------------------------------------
Dimension 12: CanteenOrderLineDIM
------------------------------------------------*/
CREATE TABLE canteenorderlinedim_0 AS
  SELECT sc_orderid,
         product_id,
         sol_lineprice
  FROM   school_canteen_orderline;
```

	SC_ORDERID	PRODUCT_ID	SOL_LINEPRICE
1	S0204	F0052	41.8
2	S0204	F0049	18.9
3	S0204	F0072	22.5
4	S0204	F0031	31.8
5	S0205	F0044	109.5

Total rows: 4346

```
/*-------------------------------------------------
Dimension 13: CateringOrderDIM
-------------------------------------------------*/
CREATE TABLE cateringorderdim_0 AS
  SELECT o.ca_orderid,
         o.ca_totalprice,
         o.ca_finalprice,
         o.ca_orderdate,
         1.0 / Count(ol.product_id)              AS ca_weight_factor,
         Listagg (ol.product_id, '_')
           within GROUP (ORDER BY ol.product_id) AS ca_product_list
  FROM   catering_order o,
         catering_orderline ol
  WHERE  o.ca_orderid = ol.ca_orderid
  GROUP  BY o.ca_orderid,
            o.ca_totalprice,
            o.ca_finalprice,
            o.ca_orderdate;
```

	CA_ORDERID	CA_TOTALPRICE	CA_FINALPRICE	CA_ORDERDATE	CA_WEIGHT_FACTOR	CA_PRODUCT_LIST
1	C0001	7.9	7.9	01/12/2018	1	F0081
2	C0002	60	67.2	11/07/2020	1	F0183
3	C0003	174.9	183.65	24/04/2019	0.2	F0029_F0060_F0076_F0096_F0182
4	C0004	110.7	118.45	11/01/2020	1	F0113
5	C0005	74.4	78.12	19/01/2018	0.25	F0010_F0099_F0162_F0163

Total rows: 1500

```
/*-------------------------------------------------
Dimension 14: CateringOrderLineDIM
-------------------------------------------------*/
CREATE TABLE cateringorderlinedim_0 AS
  SELECT ca_orderid,
         product_id,
         col_lineprice
  FROM   catering_orderline;
```

	CA_ORDERID	PRODUCT_ID	COL_LINEPRICE
1	C0200	F0081	15.8
2	C0200	F0083	39.5
3	C0200	F0064	62.7
4	C0200	F0091	13
5	C0201	F0177	520

Total rows:4494

SQL Code: Fact Tables

```
-----------------------------------------------
CanteenServiceFACT
-----------------------------------------------*/
CREATE TABLE canteenservicefact_0 AS
  SELECT o.type_id                   AS payment_type_id,
         Nvl(o.promo_code, 'no_pr') AS promo_code,
         o.status_id,
         c.type_id                   AS cust_type_id,
         s.staff_gender,
         o.sc_orderid,
         SUM(o.sc_finalprice)        AS total_sales,
         Count(o.sc_orderid)         AS total_order_number
  FROM   school_canteen_order o,
         staff s,
         customer c
  WHERE  o.staff_id = s.staff_id
         AND o.customer_id = c.customer_id
  GROUP  BY o.type_id,
            promo_code,
            o.status_id,
            c.type_id,
            s.staff_gender,
            o.sc_orderid;
```

	PAYMENT_TYPE_ID	PROMO_CODE	STATUS_ID	CUST_TYPE_ID	STAFF_GENDER	SC_ORDERID	TOTAL_SALES	TOTAL_ORDER_NUMBER
1	PT004	PR001	OS005	CT001	M	S0827	184.68	1
2	PT004	no_pr	OS005	CT003	F	S1413	53.8	1
3	PT004	PR002	OS005	CT002	M	S0636	207.91	1
4	PT001	PR003	OS005	CT001	F	S1408	112.31	1
5	PT002	no_pr	OS005	CT003	M	S0526	133.2	1

Total rows: 1500

```
/*-----------------------------------------------
CustomerFACT
-----------------------------------------------*/
CREATE TABLE customerfact_0 AS
  SELECT customer_id,
         Count(customer_id) AS total_number_of_customer
  FROM   customer
  GROUP  BY customer_id;
```

	CUSTOMER_ID	TOTAL_NUMBER_OF_CUSTOMER
1	CU047	1
2	CU011	1
3	CU019	1
4	CU045	1
5	CU029	1

Total rows: 50

```
/*-----------------------------------------------
CateringServiceFACT
-----------------------------------------------*/
CREATE TABLE cateringservicefact_0 AS
  SELECT Nvl(o.promo_code, 'no_pr')              AS promo_code,
         o.status_id,
         s.staff_gender,
         o.provider_id,
         o.ca_orderid,
         SUM(o.ca_finalprice)                    AS total_sales,
         Count(o.ca_orderid)                     AS total_order_number,
         SUM(o.ca_totalprice * p.provider_rate) AS total_delivery_cost
  FROM   catering_order o,
         staff s,
         delivery_provider p
  WHERE  o.staff_id = s.staff_id
         AND o.provider_id = p.provider_id
  GROUP  BY promo_code,
            o.status_id,
            s.staff_gender,
            o.provider_id,
            o.ca_orderid,
            o.ca_totalprice * p.provider_rate;
```

	PROMO_CODE	STATUS_ID	STAFF_GENDER	PROVIDER_ID	CA_ORDERID	TOTAL_SALES	TOTAL_ORDER_NUMBER	TOTAL_DELIVERY_COST
1	PR003	OS005	M	DP001	C0883	7.65	1	0
2	PR002	OS005	M	DP002	C0411	276.99	1	39.57
3	no_pr	OS005	M	DP002	C0262	24.03	1	3.135
4	PR002	OS005	F	DP003	C0392	145.7	1	14.57
5	no_pr	OS005	F	DP003	C0764	152.57	1	13.87

Total rows: 1500

```
/*-----------------------------------------------
ProductFACT
-----------------------------------------------*/
CREATE TABLE productfact_0 AS
  SELECT product_id,
         Count(product_id) AS total_number_of_products
  FROM   product
  GROUP  BY product_id;
```

	PRODUCT_ID	TOTAL_NUMBER_OF_PRODUCTS
1	F0001	1
2	F0004	1
3	F0015	1
4	F0019	1
5	F0023	1

Total rows: 190

Analysis Reports

Report 1

Explanation

Query Question
What is the top 3 ordered main dishes for canteen order line?

Importance of This Data
Knowing the top 3 ordered main dishes for catering, the company will have better understanding of what their market is interested in and how their market taste is like. This will enable them for market growth; what kind of menu they should add or remove from the list. This can also be the recommendation when customer asks what the favourite main dish to order is. In addition, the company can avoid ingredients waste for knowing which
dishes are ordered the most.

Level 2 OLAP QUERY

```
SELECT *
FROM (SELECT p.product_name, c.category_description, SUM(f.total_order_number) AS
total_order,
      DENSE_RANK() OVER (ORDER BY SUM(f.total_order_number) DESC) AS dense_rank
      FROM CateringServiceFACT f, CateringOrderLineDIM o, ProductDIM  p,
      ProductCategoryBridgeDIM_0 pd, CategoryDIM_0 c
      WHERE f.ca_orderid = o.ca_orderid
      AND o.product_id = p.product_id
      AND p.product_id = pd.product_id
      AND pd.category_id = c.category_id
      AND c.category_description = 'Main'
      GROUP BY p.product_name, c.category_description)
WHERE dense_rank <= 3;
```

Level 2 Output

	PRODUCT_NAME	CATEGORY_DESCRIPTION	TOTAL_ORDER	DENSE_RANK
1	Spicy Pork Bulgogi Bento	Main	56	1
2	Fried Rice	Main	55	2
3	Minced Chicken and Basil with Rice	Main	52	3

Level 0 OLAP QUERY

```
SELECT *
FROM (SELECT p.product_name, c.category_description, SUM(f.total_order_number) AS
total_order,
      DENSE_RANK() OVER (ORDER BY SUM(f.total_order_number) DESC) AS dense_rank
      FROM CateringServiceFACT_0 f, CateringOrderLineDIM_0 o, ProductDIM_0  p,
      ProductCategoryBridgeDIM_0 pd, CategoryDIM_0 c
      WHERE f.ca_orderid = o.ca_orderid
      AND o.product_id = p.product_id
      AND p.product_id = pd.product_id
      AND pd.category_id = c.category_id
      AND c.category_description = 'Main'
      GROUP BY p.product_name, c.category_description)
WHERE dense_rank <= 3;
```

Level 0 Output

	PRODUCT_NAME	CATEGORY_DESCRIPTION	TOTAL_ORDER	DENSE_RANK
1	Spicy Pork Bulgogi Bento	Main	56	1
2	Fried Rice	Main	55	2
3	Minced Chicken and Basil with Rice	Main	52	3

Report 2

Explanation

Query Question
What is the top 10% total revenue for canteen order line based on payment and customer type?

Importance of This Data
Extracting the top 10% total revenue for canteen order line based on payment and customer type, the company will have better idea of which customer type is the most profitable and what kind of payment type the customer use. The company can also implement various promotion according to the need of each customer type. Since payment types are commonly cashless, the company might be able to improve on some payment area such as the payment confirmation system.

Level 2 OLAP QUERY

```
SELECT *
FROM (
   SELECT p.type_description AS payment_type , c.type_description AS cust_type,
   SUM(f.total_sales) AS total_sales,
      PERCENT_RANK() OVER (ORDER BY SUM(f.total_sales) DESC) AS percent_rank
   FROM CanteenServiceFACT_0 f, PaymentTypeDIM_0 p, CustomerTypeDIM c
   WHERE f.payment_type_id = p.payment_type_id
   AND f.cust_type_id = c.cust_type_id
   GROUP BY p.type_description, c.type_description
)
WHERE percent_rank < 0.1;
```

Level 2 Output

	PAYMENT_TYPE	CUST_TYPE	TOTAL_SALES	PERCENT_RANK
1	Bank Transfer	School	19836	0
2	PayPal	Individual	18994,92	0,090909090909090909090909090909090909090909

Level 0 OLAP QUERY

```
SELECT *
FROM (
   SELECT p.type_description AS payment_type , c.type_description AS cust_type,
   SUM(f.total_sales) AS total_sales,
      PERCENT_RANK() OVER (ORDER BY SUM(f.total_sales) DESC) AS percent_rank
   FROM CanteenServiceFACT_0 f, PaymentTypeDIM_0 p, CustomerTypeDIM c
   WHERE f.payment_type_id = p.payment_type_id
   AND f.cust_type_id = c.cust_type_id
   GROUP BY p.type_description, c.type_description
)  WHERE percent_rank < 0.1;
```

Level 0 Output

	PAYMENT_TYPE	CUST_TYPE	TOTAL_SALES	PERCENT_RANK
1	Bank Transfer	School	19836	0
2	PayPal	Individual	18994,92	0,090909090909090909090909090909090909090909

Report 3

Explanation

Query Question
What is the dense rank of number of customer type per suburb?

Importance of This Data
Knowing which location the customers are centralized, the company can have prospect in improving the delivery system, planning on market expansion, and generating new promotion campaigns.

Level 2 OLAP QUERY

```
SELECT s.address_suburb, ct.type_description AS cust_type,
SUM(f.total_number_of_customer) AS num_of_cust,
      DENSE_RANK() OVER (ORDER BY SUM(f.total_number_of_customer) DESC)
AS dense_rank
FROM CustomerFACT f, CustomerTypeDIM ct, SuburbDIM s
WHERE f.cust_type_id = ct.cust_type_id
AND f.suburd = s.address_suburb
GROUP BY ct.type_description, s.address_suburb
ORDER BY s.address_suburb;
```

Level 2 Output

	ADDRESS_SUBURB	CUST_TYPE	NUM_OF_CUST	DENSE_RANK
1	Balaclava	Business	1	4
2	Balaclava	Individual	1	4
3	Elwood	Individual	2	3
4	Elwood	School	1	4
5	Elwood	Business	4	1
6	Melbourne	Business	1	4
7	Melbourne	Individual	2	3
8	Melbourne	School	3	2
9	Middle Park	School	2	3
10	Middle Park	Business	2	3
11	Middle Park	Individual	4	1
12	Ripponlea	Individual	4	1
13	Ripponlea	Business	2	3
14	South Melbourne	Business	1	4
15	South Melbourne	Individual	1	4
16	South Melbourne	School	4	1
17	Southbank	School	4	1
18	Southbank	Business	2	3
19	Southbank	Individual	2	3
20	St Kilda	Business	3	2
21	St Kilda	School	4	1

Level 0 OLAP QUERY

```
SELECT l.address_suburb, ct.type_description AS cust_type,
SUM(f.total_number_of_customer) AS num_of_cust,
      DENSE_RANK() OVER (ORDER BY SUM(f.total_number_of_customer) DESC)
AS dense_rank
FROM CustomerFACT_0 f, customerDIM_0 c, CustomerTypeDIM_0 ct, LocationDIM_0
l
WHERE f.customer_id = c.customer_id
AND c.cust_type_id = ct.cust_type_id
AND c.address_id = l.address_id
GROUP BY ct.type_description, l.address_suburb
ORDER BY l.address_suburb;
```

Level 0 Output

	ADDRESS_SUBURB	CUST_TYPE	NUM_OF_CUST	DENSE_RANK
1	Balaclava	Business	1	4
2	Balaclava	Individual	1	4
3	Elwood	Individual	2	3
4	Elwood	School	1	4
5	Elwood	Business	4	1
6	Melbourne	Business	1	4
7	Melbourne	School	3	2
8	Melbourne	Individual	2	3
9	Middle Park	School	2	3
10	Middle Park	Individual	4	1
11	Middle Park	Business	2	3
12	Ripponlea	Business	2	3
13	Ripponlea	Individual	4	1
14	South Melbourne	School	4	1
15	South Melbourne	Individual	1	4
16	South Melbourne	Business	1	4
17	Southbank	School	4	1
18	Southbank	Individual	2	3
19	Southbank	Business	2	3
20	St Kilda	School	4	1
21	St Kilda	Business	3	2

Report 4

Explanation

Query Question
What are the subtotals and total catering sales from each promotion, time period (month), and delivery service provider? (Cube)

Importance of This Data
The data generated by this report is helpful to analyse the sales based on a specific promotion, time period in months, and delivery service provider. This data helps the marketing team understand the best promotion that works for them, the best selling months, and the most post preferred delivery service provider in order to make executive level decision on their next strategy.

Level 2 OLAP QUERY

```
SELECT
   promo_code            AS promotion,
   the_month             AS month,
   provider_description  AS "Delivery Service Provider",
   SUM(total_sales) AS total_sales
FROM
   cateringservicefact   c
   JOIN deliveryproviderdim   d
   ON c.provider_id = d.provider_id
   JOIN timedim               t
   ON c.timeid = t.timeid
GROUP BY
   CUBE(promo_code,
        the_month,
        provider_description)
ORDER BY the_month, provider_description, promo_code;
```

Level 2 Output

	PROMOTION	MONTH	Delivery Service Provider	TOTAL_SALES
1	PR001	01	Deliveroo	1712.1
2	PR002	01	Deliveroo	1117.91
3	PR003	01	Deliveroo	1829.42
4	no_pr	01	Deliveroo	1497.55
5	(null)	01	Deliveroo	6156.98
6	PR001	01	DoorDash	556.92
7	PR002	01	DoorDash	511.3
8	PR003	01	DoorDash	1101.51
9	no_pr	01	DoorDash	1001.55
10	(null)	01	DoorDash	3171.28
11	PR001	01	Instore pick-up	1406.2
12	PR002	01	Instore pick-up	2373.48
13	PR003	01	Instore pick-up	1010.48
14	no_pr	01	Instore pick-up	1378.1
15	(null)	01	Instore pick-up	6168.26
16	PR001	01	UberEAT	2334.2
17	PR002	01	UberEAT	1267.88
18	PR003	01	UberEAT	1205.1
19	no_pr	01	UberEAT	1546.96
20	(null)	01	UberEAT	6354.14
21	PR001	01	(null)	6009.42
22	PR002	01	(null)	5270.57
23	PR003	01	(null)	5146.51

Total rows: 323

Level 0 OLAP QUERY

```
SELECT
   promo_code            AS promotion,
   to_char(ca_orderdate, 'MM') AS month,
   provider_description   AS "Delivery Service Provider",
   SUM(total_sales) AS total_sales
FROM
   cateringservicefact_0   c
   JOIN deliveryproviderdim_0   d
   ON c.provider_id = d.provider_id
   JOIN cateringorderdim_0      co
   ON c.ca_orderid = co.ca_orderid
GROUP BY
   CUBE(promo_code,
        to_char(ca_orderdate, 'MM'),
        provider_description)
ORDER By to_char(ca_orderdate, 'MM'),provider_description, promo_code;
```

Level 0 Output

	PROMOTION	MONTH	Delivery Service Provider	TOTAL_SALES
1	PR001	01	Deliveroo	1712.1
2	PR002	01	Deliveroo	1117.91
3	PR003	01	Deliveroo	1829.42
4	no_pr	01	Deliveroo	1497.55
5	(null)	01	Deliveroo	6156.98
6	PR001	01	DoorDash	556.92
7	PR002	01	DoorDash	511.3
8	PR003	01	DoorDash	1101.51
9	no_pr	01	DoorDash	1001.55
10	(null)	01	DoorDash	3171.28
11	PR001	01	Instore pick-up	1406.2
12	PR002	01	Instore pick-up	2373.48
13	PR003	01	Instore pick-up	1010.48
14	no_pr	01	Instore pick-up	1378.1
15	(null)	01	Instore pick-up	6168.26
16	PR001	01	UberEAT	2334.2
17	PR002	01	UberEAT	1267.88
18	PR003	01	UberEAT	1205.1
19	no_pr	01	UberEAT	1546.96
20	(null)	01	UberEAT	6354.14
21	PR001	01	(null)	6009.42
22	PR002	01	(null)	5270.57
23	PR003	01	(null)	5146.51

Total rows: 323

Report 5

Explanation

Query Question
What are the subtotals and total catering sales from each promotion, time period (month), and delivery service provider? (Partial Cube)

Importance of This Data
The data generated by this report is helpful to analyse the sales based on a specific promotion, time period in months, and delivery service provider. This data helps the marketing team understand the best promotion that works for them, the best selling months, and the most post preferred delivery service provider in order to make executive level decision on their next strategy.
Note: This version does not cube promo_code, but it cubes the other attributes instead.

Level 2 OLAP QUERY

```
SELECT
   promo_code            AS promotion,
   the_month             AS month,
   provider_description  AS "Delivery Service Provider",
   SUM(total_sales) AS total_sales
FROM
   cateringservicefact   c
   JOIN deliveryproviderdim   d
   ON c.provider_id = d.provider_id
   JOIN timedim               t
   ON c.timeid = t.timeid
GROUP BY
   promo_code,
   CUBE(the_month,
        provider_description)
ORDER BY the_month, provider_description, promo_code;
```

Level 2 Output

	PROMOTION	MONTH	Delivery Service Provider	TOTAL_SALES
1	PR001	01	Deliveroo	1712.1
2	PR002	01	Deliveroo	1117.91
3	PR003	01	Deliveroo	1829.42
4	no_pr	01	Deliveroo	1497.55
5	PR001	01	DoorDash	556.92
6	PR002	01	DoorDash	511.3
7	PR003	01	DoorDash	1101.51
8	no_pr	01	DoorDash	1001.55
9	PR001	01	Instore pick-up	1406.2
10	PR002	01	Instore pick-up	2373.48
11	PR003	01	Instore pick-up	1010.48
12	no_pr	01	Instore pick-up	1378.1
13	PR001	01	UberEAT	2334.2
14	PR002	01	UberEAT	1267.88
15	PR003	01	UberEAT	1205.1
16	no_pr	01	UberEAT	1546.96
17	PR001	01	(null)	6009.42
18	PR002	01	(null)	5270.57
19	PR003	01	(null)	5146.51
20	no_pr	01	(null)	5424.16
21	PR001	02	Deliveroo	2035.69
22	PR002	02	Deliveroo	2001.03
23	PR003	02	Deliveroo	559.31

Total rows: 258

Level 0 OLAP QUERY

```
SELECT
   promo_code            AS promotion,
   to_char(ca_orderdate, 'MM') AS month,
   provider_description   AS "Delivery Service Provider",
   SUM(total_sales) AS total_sales
FROM
   cateringservicefact_0   c
   JOIN deliveryproviderdim_0   d
   ON c.provider_id = d.provider_id
   JOIN cateringorderdim_0      co
   ON c.ca_orderid = co.ca_orderid
GROUP BY
   promo_code,
   CUBE(to_char(ca_orderdate, 'MM'),
        provider_description)
ORDER By to_char(ca_orderdate, 'MM'),provider_description, promo_code;
```

Level 0 Output

	PROMOTION	MONTH	Delivery Service Provider	TOTAL_SALES
1	PR001	01	Deliveroo	1712.1
2	PR002	01	Deliveroo	1117.91
3	PR003	01	Deliveroo	1829.42
4	no_pr	01	Deliveroo	1497.55
5	PR001	01	DoorDash	556.92
6	PR002	01	DoorDash	511.3
7	PR003	01	DoorDash	1101.51
8	no_pr	01	DoorDash	1001.55
9	PR001	01	Instore pick-up	1406.2
10	PR002	01	Instore pick-up	2373.48
11	PR003	01	Instore pick-up	1010.48
12	no_pr	01	Instore pick-up	1378.1
13	PR001	01	UberEAT	2334.2
14	PR002	01	UberEAT	1267.88
15	PR003	01	UberEAT	1205.1
16	no_pr	01	UberEAT	1546.96
17	PR001	01	(null)	6009.42
18	PR002	01	(null)	5270.57
19	PR003	01	(null)	5146.51
20	no_pr	01	(null)	5424.16
21	PR001	02	Deliveroo	2035.69
22	PR002	02	Deliveroo	2001.03
23	PR003	02	Deliveroo	559.31

Total rows: 258

Report 6

Explanation

Query Question
What are the total weekly days sales canteen sales per month?

Importance of This Data
This Data is important to help the marketing team understand and evaluate which daily week (Monday…Sunday) results in the most sales, so they can focus their marketing strategies on other less profitable days. Perhaps this can also allow HR to staff the kitchen according to the business, which can be estimated from the total sales on that day.

Level 2 OLAP QUERY

```
SELECT
   the_month   AS month,
   weeklyday   AS day,
   SUM(total_sales) AS total_sales
FROM
   canteenservicefact   c
   JOIN timedim             t
   ON c.timeid = t.timeid
GROUP BY
   ROLLUP(the_month,
         weeklyday)
ORDER BY
   the_month;
```

Level 2 Output

	MONTH	DAY	TOTAL_SALES
1	01	Friday	3349.5
2	01	Monday	3030.01
3	01	Saturday	3088.25
4	01	Sunday	2284.02
5	01	Thursday	2786.48
6	01	Tuesday	2267.32
7	01	Wednesday	3503.72
8	01	(null)	20309.3
9	02	Friday	2795.48
10	02	Monday	4273.05
11	02	Saturday	3061.79
12	02	Sunday	3408.74
13	02	Thursday	1292.02
14	02	Tuesday	2561.84
15	02	Wednesday	2054.95
16	02	(null)	19447.87
17	03	Friday	2176.16
18	03	Monday	3470.88
19	03	Saturday	2142.01
20	03	Sunday	4208.42
21	03	Thursday	4314.24
22	03	Tuesday	3813.32
23	03	Wednesday	3682.7

Total rows: 97

Level 0 OLAP QUERY

```
SELECT
   to_char(sc_orderdate, 'MM') AS month,
   to_char(sc_orderdate, 'Day') AS day,
   SUM(total_sales) AS total_sales
FROM
   canteenservicefact_0   c
   JOIN canteenorderdim_0      co
   ON c.sc_orderid = co.sc_orderid
GROUP BY
   ROLLUP(to_char(sc_orderdate, 'MM'),
         to_char(sc_orderdate, 'Day'))
ORDER BY
   to_char(sc_orderdate, 'MM');
```

Level 0 Output

	MONTH	DAY	TOTAL_SALES
1	01	Friday	3349.5
2	01	Monday	3030.01
3	01	Saturday	3088.25
4	01	Sunday	2284.02
5	01	Thursday	2786.48
6	01	Tuesday	2267.32
7	01	Wednesday	3503.72
8	01	(null)	20309.3
9	02	Friday	2795.48
10	02	Monday	4273.05
11	02	Saturday	3061.79
12	02	Sunday	3408.74
13	02	Thursday	1292.02
14	02	Tuesday	2561.84
15	02	Wednesday	2054.95
16	02	(null)	19447.87
17	03	Friday	2176.16
18	03	Monday	3470.88
19	03	Saturday	2142.01
20	03	Sunday	4208.42
21	03	Thursday	4314.24
22	03	Tuesday	3813.32
23	03	Wednesday	3682.7

Total rows: 97

Report 7

Explanation

Query Question
What is the monthly total delivery cost per provider?

Explanation on Importance of This Data
This data tells the viewer about the total cost per provider on monthly basis, and this can be relevant for the financial department to understand how much of their sales is going to delivery funds, and perhaps cut down on expensive delivery providers or renegotiate the rate if the ratio of the delivery cost to the total profit is off the dedicated margin.

Level 2 OLAP QUERY

```
SELECT
   the_month            AS month,
   provider_description   AS "Delivery Provider",
   SUM(total_delivery_cost) AS total_sales
FROM
   cateringservicefact   c
   JOIN timedim             t
   ON c.timeid = t.timeid
   JOIN deliveryproviderdim   d
   ON d.provider_id = c.provider_id
GROUP BY
   the_month,
   ROLLUP(provider_description)
ORDER BY
   the_month;
```

Level 2 Output

	MONTH	Delivery Provider	TOTAL_SALES
1	01	Deliveroo	710.304
2	01	DoorDash	311.17
3	01	Instore pick-up	0
4	01	UberEAT	881.97
5	01	(null)	1903.444
6	02	Deliveroo	644.064
7	02	DoorDash	482.58
8	02	Instore pick-up	0
9	02	UberEAT	820.59
10	02	(null)	1947.234
11	03	Deliveroo	609.924
12	03	DoorDash	562.22
13	03	Instore pick-up	0
14	03	UberEAT	930.255
15	03	(null)	2102.399
16	04	Deliveroo	442.464
17	04	DoorDash	608.36
18	04	Instore pick-up	0
19	04	UberEAT	570.9
20	04	(null)	1621.724
21	05	Deliveroo	587.688
22	05	DoorDash	499.07
23	05	Instore pick-up	0

Total rows: 60

Level 0 OLAP QUERY

```
SELECT
   to_char(ca_orderdate, 'MM') AS month,
```

```
    provider_description AS "Delivery Provider",
    SUM(total_delivery_cost) AS total_sales
FROM
    cateringservicefact_0   c
    JOIN cateringorderdim_0      co
    ON c.ca_orderid = co.ca_orderid
    JOIN deliveryproviderdim_0   d
    ON d.provider_id = c.provider_id
GROUP BY
    to_char(ca_orderdate, 'MM'),
    ROLLUP(provider_description)
ORDER BY
    to_char(ca_orderdate, 'MM');
```

Level 0 Output

	MONTH	Delivery Provider	TOTAL_SALES
1	01	Deliveroo	710.304
2	01	DoorDash	311.17
3	01	Instore pick-up	0
4	01	UberEAT	881.97
5	01	(null)	1903.444
6	02	Deliveroo	644.064
7	02	DoorDash	482.58
8	02	Instore pick-up	0
9	02	UberEAT	820.59
10	02	(null)	1947.234
11	03	Deliveroo	609.924
12	03	DoorDash	562.22
13	03	Instore pick-up	0
14	03	UberEAT	930.255
15	03	(null)	2102.399
16	04	Deliveroo	442.464
17	04	DoorDash	608.36
18	04	Instore pick-up	0
19	04	UberEAT	570.9
20	04	(null)	1621.724
21	05	Deliveroo	587.688
22	05	DoorDash	499.07
23	05	Instore pick-up	0

Total rows: 60

Report 8 Moving

Explanation

Query Question
What are the total catering sales and moving catering sales of 3 yearly of Savoury dishes in each year?

Explanation on Importance of This Data
Having a smoother (less noise) data of the total revenue in each year will ease the company in analysing and determining trends in the data. From this data, the company can understand how the selling of savoury dishes in catering order line has performed and predict what might happen to it in the future. This will also give the company idea to change things (e.g. more savoury dishes, improvement on flavours, others) to move their catering business in the right direction.

Level 2 OLAP QUERY

```
SELECT the_year, SUM(final_sales) AS total_revenue,
TO_CHAR(AVG(SUM(final_sales)) OVER(ORDER BY the_year ROWS 2
PRECEDING), '9,999,999.99') AS moving_revenue
FROM (
      SELECT t.the_year, f.total_sales, c.ca_totalprice,
      (SUM(co.col_lineprice)/c.ca_totalprice) AS percentage,
(f.total_sales*(SUM(co.col_lineprice)/c.ca_totalprice)) AS final_sales
      FROM CateringServiceFact f, TimeDIM t, CateringOrderDIM c,
CateringOrderLineDIM co,
      ProductCategoryBridgeDIM p, CategoryDIM ct
      WHERE f.timeID = t.timeID
      AND f.ca_orderid = c.ca_orderid
      AND c.ca_orderid = co.ca_orderid
      AND co.product_id = p.product_id
      AND p.category_id = ct.category_id
      AND ct.category_description = 'Savoury'
      GROUP BY t.the_year, f.total_sales, c.ca_totalprice)
GROUP BY the_year;
```

To calculate the final price of savoury dishes only, we first calculate the proportion (percentage) of savoury dishes in an order by dividing line price over total price. Then, we multiply the proportion with the total sales (final price) to obtain the final price of savoury dishes only.

Level 2 Output

	THE_YEAR	TOTAL_REVENUE	MOVING_REVENUE
1	2018	40238,321717356127384634593574972993660 2	40,238.32
2	2019	40202,245544553325292426785751939042320 8	40,220.28
3	2020	48492,421811679443016022234986115427707 4	42,977.66

Level 0 OLAP QUERY

```
SELECT the_year, SUM(final_sales) AS total_revenue,
TO_CHAR(AVG(SUM(final_sales)) OVER(ORDER BY the_year ROWS 2
PRECEDING), '9,999,999.99') AS moving_revenue
FROM (
      SELECT TO_CHAR(c.ca_orderdate,'YYYY') AS the_year, f.total_sales,
c.ca_totalprice,
      (SUM(co.col_lineprice)/c.ca_totalprice) AS percentage,
(f.total_sales*(SUM(co.col_lineprice)/c.ca_totalprice)) AS final_sales
      FROM CateringServiceFACT_0 f, CateringOrderDIM_0 c, CateringOrderLineDIM_0
co, ProductCategoryBridgeDIM_0 p, CategoryDIM_0 ct
      WHERE f.ca_orderid = c.ca_orderid
      AND c.ca_orderid = co.ca_orderid
      AND co.product_id = p.product_id
      AND p.category_id = ct.category_id
      AND ct.category_description = 'Savoury'
      GROUP BY c.ca_orderdate, f.total_sales, c.ca_totalprice)
GROUP BY the_year;
```

To calculate the final price of savoury dishes only, we first calculate the proportion (percentage) of savoury dishes in an order by dividing line price over total price. Then, we multiply the proportion with the total sales (final price) to obtain the final price of savoury dishes only.

Level 0 Output

	THE_YEAR	TOTAL_REVENUE	MOVING_REVENUE
1	2018	40238,321717356127384634593574972993660 3	40,238.32
2	2019	40202,245544553325292426785751939042320 7	40,220.28
3	2020	48492,421811679443016022234986115427707 6	42,977.66

Report 8 Cumulative

Explanation

Query Question

What are the total catering sales and cumulative total catering sales of Savoury dishes in each year?

Explanation on Importance of This Data

Looking at the cumulative total catering sales, the company can gain insight about the growth of the sold savoury dishes in catering order line from year to year. Gaining this information will also give the company better understanding on how to improve its strategy in selling and improving savoury dishes.

Level 2 OLAP QUERY

```
SELECT the_year, SUM(final_sales) AS total_revenue,
TO_CHAR(SUM(SUM(final_sales)) OVER(ORDER BY the_year ROWS UNBOUNDED
PRECEDING), '9,999,999.99') AS cummulative_revenue
FROM (
      SELECT t.the_year, f.total_sales, c.ca_totalprice,
(SUM(co.col_lineprice)/c.ca_totalprice) AS percentage,
      (f.total_sales*(SUM(co.col_lineprice)/c.ca_totalprice)) AS final_sales
      FROM CateringServiceFact f, TimeDIM t, CateringOrderDIM c,
CateringOrderLineDIM co,
      ProductCategoryBridgeDIM p, CategoryDIM ct
      WHERE f.timeID = t.timeID
      AND f.ca_orderid = c.ca_orderid
      AND c.ca_orderid = co.ca_orderid
      AND co.product_id = p.product_id
      AND p.category_id = ct.category_id
      AND ct.category_description = 'Savoury'
      GROUP BY t.the_year, f.total_sales, c.ca_totalprice)
GROUP BY the_year;
```

To calculate the final price of savoury dishes only, we first calculate the proportion (percentage) of savoury dishes in an order by dividing line price over total price. Then, we multiply the proportion with the total sales (final price) to obtain the final price of savoury dishes only.

Level 2 Output

	THE_YEAR	TOTAL_REVENUE	CUMULATIVE_REVENUE
1	2018	40238,321717356127384634593574972993660 2	40,238.32
2	2019	40202,245544553325292426785751939042320 8	80,440.57
3	2020	48492,421811679443016022234986115427707 4	128,932.99

Level 0 OLAP QUERY

```
SELECT the_year, SUM(final_sales) AS total_revenue,
TO_CHAR(SUM(SUM(final_sales)) OVER(ORDER BY the_year ROWS UNBOUNDED
PRECEDING), '9,999,999.99') AS cummulative_revenue
FROM (
   SELECT TO_CHAR(c.ca_orderdate,'YYYY') AS the_year, f.total_sales,
c.ca_totalprice,
   (SUM(co.col_lineprice)/c.ca_totalprice) AS percentage,
   (f.total_sales*(SUM(co.col_lineprice)/c.ca_totalprice)) AS final_sales
   FROM CateringServiceFACT_0 f,MCateringOrderDIM_0 c, CateringOrderLineDIM_0
co, ProductCategoryBridgeDIM_0 p, CategoryDIM_0 ct
   WHERE f.ca_orderid = c.ca_orderid
   AND c.ca_orderid = co.ca_orderid
   AND co.product_id = p.product_id
   AND p.category_id = ct.category_id
   AND ct.category_description = 'Savoury'
   GROUP BY c.ca_orderdate, f.total_sales, c.ca_totalprice)
GROUP BY the_year;
```

To calculate the final price of savoury dishes only, we first calculate the proportion (percentage) of savoury dishes in an order by dividing line price over total price. Then, we multiply the proportion with the total sales (final price) to obtain the final price of savoury dishes only.

Level 0 Output

	THE_YEAR	TOTAL_REVENUE	CUMULATIVE_REVENUE
1	2018	40238,321717356127384634593574972993660 3	40,238.32
2	2019	40202,245544553325292426785751939042320 7	80,440.57
3	2020	48492,421811679443016022234986115427707 6	128,932.99

Report 9 Moving

Explanation

Query Question

What are the total and moving catering promotion's value of 3 monthly?

Importance of This Data

Moving aggregate is used to smooth out some outlier figures in certain period of time which cause in easier trends analysis and decision. From moving average data on catering promotion's value, the company can understand the trend of promotion's usage of 3 monthly. The company can also gain idea of how much total promotion value used by the customers. Last but not least, the company will gain insight on how to change or keep the promotion to move their catering business in the right direction.

Level 2 OLAP QUERY

```
SELECT time_id, SUM(promotion_val) AS promotion_val,
   TO_CHAR(AVG(SUM(promotion_val)) OVER(ORDER BY time_id ROWS 2
PRECEDING), '9,999,999.99') AS moving_3_month
FROM (
      SELECT (t.the_year || t.the_month) AS time_id,
      f.total_sales, c.ca_totalprice, f.total_delivery_cost,
      (c.ca_totalprice + f.total_delivery_cost - f.total_sales) AS promotion_val
      FROM CateringServiceFACT f, CateringOrderDIM c, DeliveryProviderDIM d,
TimeDIM t
      WHERE f.ca_orderid = c.ca_orderid
      AND f.timeID = t.timeID
      AND f.provider_id = d.provider_id
      AND f.promo_code != 'no_pr'
      GROUP BY t.the_year, t.the_month, f.total_sales, c.ca_totalprice,
f.total_delivery_cost, f.ca_orderid)
```

To calculate the promotion value, the total price will be added with total delivery cost from fact table then it is subtracted with the final price (total_sales). In level 2, order ID is used in group by expression to ensure all orders are included in the calculation.

Level 2 Output

	TIME_ID	PROMOTION_VAL	MOVING_3_MONTH
1	201801	340,838	340.84
2	201802	480,943	410.89
3	201803	495,612	439.13
4	201804	510,671	495.74
5	201805	308,968	438.42
6	201806	554,37	458.00
7	201807	395,511	419.62
8	201808	355,446	435.11
9	201809	753,254	501.40
10	201810	417,537	508.75
11	201811	543,974	571.59
12	201812	579,449	513.65
13	201901	339,51	487.64
14	201902	682,087	533.68
15	201903	602,765	541.45
16	201904	348,634	544.50
17	201905	465,4	472.27
18	201906	504,854	439.63
19	201907	608,823	526.36
20	201908	372,901	495.53
21	201909	566,029	515.92
22	201910	350,367	429.77
23	201911	562,704	493.03
24	201912	284,94	399.34
25	202001	967,414	605.02
26	202002	471,838	574.73
27	202003	765,2	734.82
28	202004	847,077	694.71
29	202005	741,483	784.59
30	202006	817,147	801.90
31	202007	800,063	786.23
32	202008	890,116	835.78

Level 0 OLAP QUERY

```
SELECT time_id, SUM(promotion_val) AS promotion_val,
   TO_CHAR(AVG(SUM(promotion_val)) OVER(ORDER BY time_id ROWS 2
PRECEDING), '9,999,999.99') AS moving_3_month
FROM (
      SELECT (TO_CHAR(c.ca_orderdate, 'YYYY') || TO_CHAR(c.ca_orderdate, 'MM'))
AS time_id,
      f.total_sales, c.ca_totalprice, f.total_delivery_cost,
      (c.ca_totalprice + f.total_delivery_cost - f.total_sales) AS promotion_val
      FROM CateringServiceFACT_0 f, CateringOrderDIM_0 c, DeliveryProviderDIM_0
d
      WHERE f.ca_orderid = c.ca_orderid
      AND f.provider_id = d.provider_id
      AND f.promo_code != 'no_pr'
      GROUP BY c.ca_orderdate, f.total_sales, c.ca_totalprice, f.total_delivery_cost)
GROUP BY time_id;
```

To calculate the promotion value, the total price will be added with total delivery cost from fact table then it is subtracted with the final price (total_sales).

Level 0 Output

	TIME_ID	PROMOTION_VAL	MOVING_3_MONTH
1	201801	340,838	340.84
2	201802	480,943	410.89
3	201803	495,612	439.13
4	201804	510,671	495.74
5	201805	308,968	438.42
6	201806	554,37	458.00
7	201807	395,511	419.62
8	201808	355,446	435.11
9	201809	753,254	501.40
10	201810	417,537	508.75
11	201811	543,974	571.59
12	201812	579,449	513.65
13	201901	339,51	487.64
14	201902	682,087	533.68
15	201903	602,765	541.45
16	201904	348,634	544.50
17	201905	465,4	472.27
18	201906	504,854	439.63
19	201907	608,823	526.36
20	201908	372,901	495.53
21	201909	566,029	515.92
22	201910	350,367	429.77
23	201911	562,704	493.03
24	201912	284,94	399.34
25	202001	967,414	605.02
26	202002	471,838	574.73
27	202003	765,2	734.82
28	202004	847,077	694.71
29	202005	741,483	784.59
30	202006	817,147	801.90
31	202007	800,063	786.23
32	202008	890,116	835.78

Report 9 Cumulative

Explanation

Query Question

What are the total and cumulative catering promotion's value in each month-year?

Importance of This Data

Looking at the cumulative total of promotion in catering order line, the company can gain insight on how much the cumulative of promotion value is used each month. Gaining this information will also give the company better understanding on how to improve its marketing strategy in applying promotion.

Level 2 OLAP QUERY

```
SELECT time_id, SUM(promotion_val) AS promotion_val,
   TO_CHAR(SUM(SUM(promotion_val)) OVER(ORDER BY time_id ROWS
UNBOUNDED PRECEDING), '9,999,999.99') AS cumulative_promotion_val
FROM (
      SELECT (t.the_year || t.the_month) AS time_id,
      f.total_sales, c.ca_totalprice, f.total_delivery_cost,
      (c.ca_totalprice + f.total_delivery_cost - f.total_sales) AS promotion_val
      FROM CateringServiceFACT f, CateringOrderDIM c, DeliveryProviderDIM d,
TimeDIM t
      WHERE f.ca_orderid = c.ca_orderid
      AND f.timeID = t.timeID
      AND f.provider_id = d.provider_id
      AND f.promo_code != 'no_pr'
      GROUP BY t.the_year, t.the_month, f.total_sales, c.ca_totalprice,
f.total_delivery_cost, f.ca_orderid)
GROUP BY time_id;
```

To calculate the promotion value, the total price will be added with total delivery cost from fact table then it is subtracted with the final price (total_sales). In level 2, order ID is used in group by expression to ensure all orders are included in the calculation.

Level 2 Output

	TIME_ID	PROMOTION_VAL	CUMULATIVE_PROMOTION_VAL
1	201801	340,838	340.84
2	201802	480,943	821.78
3	201803	495,612	1,317.39
4	201804	510,671	1,828.06
5	201805	308,968	2,137.03
6	201806	554,37	2,691.40
7	201807	395,511	3,086.91
8	201808	355,446	3,442.36
9	201809	753,254	4,195.61
10	201810	417,537	4,613.15
11	201811	543,974	5,157.12
12	201812	579,449	5,736.57
13	201901	339,51	6,076.08
14	201902	682,087	6,758.17
15	201903	602,765	7,360.94
16	201904	348,634	7,709.57
17	201905	465,4	8,174.97
18	201906	504,854	8,679.82
19	201907	608,823	9,288.65
20	201908	372,901	9,661.55
21	201909	566,029	10,227.58
22	201910	350,367	10,577.94
23	201911	562,704	11,140.65
24	201912	284,94	11,425.59
25	202001	967,414	12,393.00
26	202002	471,838	12,864.84
27	202003	765,2	13,630.04
28	202004	847,077	14,477.12
29	202005	741,483	15,218.60
30	202006	817,147	16,035.75
31	202007	800,063	16,835.81
32	202008	890,116	17,725.93

Level 0 OLAP QUERY

```
SELECT time_id, SUM(promotion_val) AS promotion_val,
   TO_CHAR(SUM(SUM(promotion_val)) OVER(ORDER BY time_id ROWS
UNBOUNDED PRECEDING), '9,999,999.99') AS cumulative_promotion_val
FROM (
      SELECT (TO_CHAR(c.ca_orderdate, 'YYYY') || TO_CHAR(c.ca_orderdate, 'MM'))
AS time_id,
      f.total_sales, c.ca_totalprice, f.total_delivery_cost,
      (c.ca_totalprice + f.total_delivery_cost - f.total_sales) AS promotion_val
      FROM CateringServiceFACT_0 f, CateringOrderDIM_0 c, DeliveryProviderDIM_0
d
      WHERE f.ca_orderid = c.ca_orderid
      AND f.provider_id = d.provider_id
      AND f.promo_code != 'no_pr'
      GROUP BY c.ca_orderdate, f.total_sales, c.ca_totalprice, f.total_delivery_cost)
GROUP BY time_id;
```

Level 0 Output

	TIME_ID	PROMOTION_VAL	CUMULATIVE_PROMOTION_VAL
1	201801	340,838	340.84
2	201802	480,943	821.78
3	201803	495,612	1,317.39
4	201804	510,671	1,828.06
5	201805	308,968	2,137.03
6	201806	554,37	2,691.40
7	201807	395,511	3,086.91
8	201808	355,446	3,442.36
9	201809	753,254	4,195.61
10	201810	417,537	4,613.15
11	201811	543,974	5,157.12
12	201812	579,449	5,736.57
13	201901	339,51	6,076.08
14	201902	682,087	6,758.17
15	201903	602,765	7,360.94
16	201904	348,634	7,709.57
17	201905	465,4	8,174.97
18	201906	504,854	8,679.82
19	201907	608,823	9,288.65
20	201908	372,901	9,661.55
21	201909	566,029	10,227.58
22	201910	350,367	10,577.94
23	201911	562,704	11,140.65
24	201912	284,94	11,425.59
25	202001	967,414	12,393.00
26	202002	471,838	12,864.84
27	202003	765,2	13,630.04
28	202004	847,077	14,477.12
29	202005	741,483	15,218.60
30	202006	817,147	16,035.75
31	202007	800,063	16,835.81
32	202008	890,116	17,725.93

Report 10 Moving

Explanation

Query Question
What are the total and moving number of canteen's order of 3 monthly?

Importance of This Data
Having the average order of 3 monthly at each month from the start of the canteen business will make the company understand the order trend. Referring to this data, the company can also predict things such as the month of their order peak.

Level 2 OLAP QUERY

```
SELECT (t.the_year || t.the_month) AS time_id,
      SUM(f.total_order_number) AS total_order_number,
      TO_CHAR(AVG(SUM(f.total_order_number))
      OVER (ORDER BY (t.the_year || t.the_month) ROWS 2 PRECEDING),
'9,999,999.99') AS moving_3_months
FROM CanteenServiceFACT f, CanteenOrderDIM c, TimeDIM t
WHERE f.sc_orderid = c.sc_orderid
AND f.timeID = t.timeID
GROUP BY t.the_year, t.the_month;
```

Level 2 Output

	TIME_ID	TOTAL_ORDER_NUMBER	MOVING_3_MONTHS
1	201901	63	63.00
2	201902	52	57.50
3	201903	56	57.00
4	201904	62	56.67
5	201905	59	59.00
6	201906	67	62.67
7	201907	65	63.67
8	201908	80	70.67
9	201909	65	70.00
10	201910	61	68.67
11	201911	54	60.00
12	201912	68	61.00
13	202001	86	69.33
14	202002	97	83.67
15	202003	91	91.33
16	202004	90	92.67
17	202005	91	90.67
18	202006	91	90.67
19	202007	112	98.00
20	202008	90	97.67

Level 0 OLAP QUERY

```
SELECT (TO_CHAR(c.sc_orderdate, 'YYYY') || TO_CHAR(c.sc_orderdate, 'MM')) AS
time_id,
      SUM(f.total_order_number) AS total_order_number,
      TO_CHAR(AVG(SUM(f.total_order_number)) OVER
      (ORDER BY (TO_CHAR(c.sc_orderdate, 'YYYY') || TO_CHAR(c.sc_orderdate,
'MM')) ROWS 2 PRECEDING), '9,999,999.99') AS moving_3_months
FROM CanteenServiceFACT_0 f, CanteenOrderDIM_0 c
WHERE f.sc_orderid = c.sc_orderid
GROUP BY (TO_CHAR(c.sc_orderdate, 'YYYY') || TO_CHAR(c.sc_orderdate, 'MM'));
```

Level 0 Output

	TIME_ID	TOTAL_ORDER_NUMBER	MOVING_3_MONTHS
1	201901	63	63.00
2	201902	52	57.50
3	201903	56	57.00
4	201904	62	56.67
5	201905	59	59.00
6	201906	67	62.67
7	201907	65	63.67
8	201908	80	70.67
9	201909	65	70.00
10	201910	61	68.67
11	201911	54	60.00
12	201912	68	61.00
13	202001	86	69.33
14	202002	97	83.67
15	202003	91	91.33
16	202004	90	92.67
17	202005	91	90.67
18	202006	91	90.67
19	202007	112	98.00
20	202008	90	97.67

Report 10 Cumulative

Explanation

Query Question
What are the total and cumulative number of canteen's order in each month-year?

Explanation on Importance of This Data
Getting data on the number of total and cumulative number of canteen's order, the company can gain information on their canteen business growth from the start. They can see how much their company has grown from the start to the last date the data retrieved. Having this information, the company can determine whether to continue or stop the business, and can improve things (e.g. market expansion, marketing strategy, payment and/or delivery system, others).

Level 2 OLAP QUERY

```
SELECT (t.the_year || t.the_month) AS time_id,
       SUM(f.total_order_number) AS total_order_number,
       TO_CHAR(SUM(SUM(f.total_order_number))
       OVER (ORDER BY (t.the_year || t.the_month) ROWS UNBOUNDED
PRECEDING),
       '9,999,999.99') AS cumulative_order_number
FROM CanteenServiceFACT f, CanteenOrderDIM c, TimeDIM t
WHERE f.sc_orderid = c.sc_orderid
AND f.timeID = t.timeID
GROUP BY t.the_year, t.the_month;
```

Level 2 Output

	TIME_ID	TOTAL_ORDER_NUMBER	CUMULATIVE_ORDER_NUMBER
1	201901	63	63.00
2	201902	52	115.00
3	201903	56	171.00
4	201904	62	233.00
5	201905	59	292.00
6	201906	67	359.00
7	201907	65	424.00
8	201908	80	504.00
9	201909	65	569.00
10	201910	61	630.00
11	201911	54	684.00
12	201912	68	752.00
13	202001	86	838.00
14	202002	97	935.00
15	202003	91	1,026.00
16	202004	90	1,116.00
17	202005	91	1,207.00
18	202006	91	1,298.00
19	202007	112	1,410.00
20	202008	90	1,500.00

Level 0 OLAP QUERY

```
SELECT (TO_CHAR(c.sc_orderdate, 'YYYY') || TO_CHAR(c.sc_orderdate, 'MM')) AS
time_id,
      SUM(f.total_order_number) AS total_order_number,
      TO_CHAR(SUM(SUM(f.total_order_number)) OVER
      (ORDER BY (TO_CHAR(c.sc_orderdate, 'YYYY') || TO_CHAR(c.sc_orderdate,
'MM')) ROWS UNBOUNDED PRECEDING),  '9,999,999.99') AS
cumulative_order_number
FROM CanteenServiceFACT_0 f, CanteenOrderDIM_0 c
WHERE f.sc_orderid = c.sc_orderid
GROUP BY (TO_CHAR(c.sc_orderdate, 'YYYY') || TO_CHAR(c.sc_orderdate, 'MM'));
```

Level 0 Output

	TIME_ID	TOTAL_ORDER_NUMBER	CUMULATIVE_ORDER_NUMBER
1	201901	63	63.00
2	201902	52	115.00
3	201903	56	171.00
4	201904	62	233.00
5	201905	59	292.00
6	201906	67	359.00
7	201907	65	424.00
8	201908	80	504.00
9	201909	65	569.00
10	201910	61	630.00
11	201911	54	684.00
12	201912	68	752.00
13	202001	86	838.00
14	202002	97	935.00
15	202003	91	1,026.00
16	202004	90	1,116.00
17	202005	91	1,207.00
18	202006	91	1,298.00
19	202007	112	1,410.00
20	202008	90	1,500.00

Report 11

Explanation

Query Question
Show ranking of each cuisine based on the monthly total number of sales for school canteen orders and the ranking of each customer type based on the monthly total number of sales for school canteen orders.

Explanation on Importance of This Data
This data is relevant to understand the most preferred type of cuisine per customer type on monthly basis. This can help the kitchen management to focus on which cuisine to keep pushing, and on which cuisine to cut down on, to avoid financial loss due to the lack of order for certain cuisines. For example, if Korean cuisine is the most ordered, then stock up all the ingredients that make the Korean dishes. On the other hand, if Thai cuisine is the least ordered, then avoid over stocking the relevant ingredients for that.

Level 2 OLAP QUERY

```
SELECT
   category_description    AS cuisine,
   cust.type_description   AS customer_type,
   the_month               AS month,
   SUM(total_order_number) AS "Monthly Total Number of Sales",
   RANK() OVER(
      PARTITION BY category_description
      ORDER BY
         SUM(total_order_number) DESC
   ) AS rank_by_cuisine,
   RANK() OVER(
      PARTITION BY cust.type_description
      ORDER BY
         SUM(total_order_number) DESC
   ) AS rank_by_customer_type
FROM
   canteenservicefact        c
   JOIN timedim                  t
   ON c.timeid = t.timeid
   JOIN customertypedim          cust
   ON cust.cust_type_id = c.cust_type_id
   JOIN canteenorderdim          co
   ON c.sc_orderid = co.sc_orderid
   JOIN canteenorderlinedim      col
   ON co.sc_orderid = col.sc_orderid
   JOIN productdim               p
   ON p.product_id = col.product_id
   JOIN productcategorybridgedim   pc
   ON p.product_id = pc.product_id
   JOIN categorydim              cate
   ON cate.category_id = pc.category_id
WHERE
```

```
    category_type = 'cuisine'
GROUP BY
    category_description,
    cust.type_description,
    the_month
ORDER BY
    category_description,
    rank_by_cuisine,
    rank_by_customer_type;
```

Level 2 Output

	CUISINE	CUSTOMER_TYPE	MONTH	Monthly Total Number of Sales	RANK_BY_CUISINE	RANK_BY_CUSTOMER_TYPE
1	Indonesian	School	08	28	1	22
2	Indonesian	School	04	27	2	23
3	Indonesian	School	07	27	2	23
4	Indonesian	Individual	07	26	4	20
5	Indonesian	Individual	08	25	5	22
6	Indonesian	Individual	04	24	6	23
7	Indonesian	Business	04	23	7	19
8	Indonesian	Business	03	23	7	19
9	Indonesian	Individual	01	23	7	25
10	Indonesian	Business	05	21	10	21
11	Indonesian	School	06	21	10	27
12	Indonesian	Individual	06	20	12	28
13	Indonesian	School	03	20	12	28
14	Indonesian	Business	08	19	14	23
15	Indonesian	Business	01	17	15	28
16	Indonesian	Business	07	17	15	28
17	Indonesian	Business	06	17	15	28
18	Indonesian	Individual	02	17	15	29
19	Indonesian	Individual	05	17	15	29
20	Indonesian	Individual	09	15	20	32
21	Indonesian	Individual	03	15	20	32
22	Indonesian	Business	02	15	20	32
23	Indonesian	School	01	14	23	30

Total rows: 108

Level 0 OLAP QUERY

```
SELECT
   category_description    AS cuisine,
   cust.type_description   AS customer_type,
   to_char(sc_orderdate, 'MM') AS month,
   SUM(total_order_number) AS "Monthly Total Number of Sales",
   RANK() OVER(
      PARTITION BY category_description
      ORDER BY
         SUM(total_order_number) DESC
   ) AS rank_by_cuisine,
   RANK() OVER(
      PARTITION BY cust.type_description
      ORDER BY
         SUM(total_order_number) DESC
   ) AS rank_by_customer_type
FROM
   canteenservicefact_0        c
   JOIN customertypedim_0           cust
   ON cust.cust_type_id = c.cust_type_id
   JOIN canteenorderdim_0           co
   ON c.sc_orderid = co.sc_orderid
   JOIN canteenorderlinedim_0       col
   ON co.sc_orderid = col.sc_orderid
   JOIN productdim_0                p
   ON p.product_id = col.product_id
   JOIN productcategorybridgedim_0   pc
   ON p.product_id = pc.product_id
   JOIN categorydim_0               cate
   ON cate.category_id = pc.category_id
WHERE
   category_type = 'cuisine'
GROUP BY
   category_description,
   cust.type_description,
   to_char(sc_orderdate, 'MM')
ORDER BY
   category_description,
   rank_by_cuisine,
   rank_by_customer_type;
```

Level 0 Output

	CUISINE	CUSTOMER_TYPE	MONTH	Monthly Total Number of Sales	RANK_BY_CUISINE	RANK_BY_CUSTOMER_TYPE
1	Indonesian	School	08	28	1	22
2	Indonesian	School	04	27	2	23
3	Indonesian	School	07	27	2	23
4	Indonesian	Individual	07	26	4	20
5	Indonesian	Individual	08	25	5	22
6	Indonesian	Individual	04	24	6	23
7	Indonesian	Business	04	23	7	19
8	Indonesian	Business	03	23	7	19
9	Indonesian	Individual	01	23	7	25
10	Indonesian	Business	05	21	10	21
11	Indonesian	School	06	21	10	27
12	Indonesian	Individual	06	20	12	28
13	Indonesian	School	03	20	12	28
14	Indonesian	Business	08	19	14	23
15	Indonesian	Business	01	17	15	28
16	Indonesian	Business	07	17	15	28
17	Indonesian	Business	06	17	15	28
18	Indonesian	Individual	02	17	15	29
19	Indonesian	Individual	05	17	15	29
20	Indonesian	Individual	09	15	20	32
21	Indonesian	Individual	03	15	20	32
22	Indonesian	Business	02	15	20	32
23	Indonesian	School	01	14	23	30

Total rows: 108

Report 12

Explanation

Query Question
What is the cumulative total school canteen sales for each meal based on each month?

Explanation on Importance of This Data
This data is relevant to the kitchen management to understand how much revenue, cumulatively, they are making from each meal on monthly basis. This can help them focus on which meals to continue pushing each month, and on what other meals should they cut down on producing to avoid financial loss or compromised profit.

Level 2 OLAP QUERY

```
SELECT
    category_description   AS meal,
    the_month              AS month,
    to_char(SUM(total_sales *(sol_lineprice / sc_totalprice)), '9,999,999,999.99'
    ) AS "Monthly Total Sales",
    to_char(SUM(SUM(total_sales *(sol_lineprice / sc_totalprice))) OVER(
        PARTITION BY category_description
        ORDER BY
            category_description, the_month
        ROWS UNBOUNDED PRECEDING
    ), '9,999,999,999.99') AS cum_sales
FROM
    canteenservicefact        c
    JOIN timedim                  t
    ON c.timeid = t.timeid
    JOIN customertypedim          cust
    ON cust.cust_type_id = c.cust_type_id
    JOIN canteenorderdim          co
    ON c.sc_orderid = co.sc_orderid
    JOIN canteenorderlinedim      col
    ON co.sc_orderid = col.sc_orderid
    JOIN productdim               p
    ON p.product_id = col.product_id
    JOIN productcategorybridgedim   pc
    ON p.product_id = pc.product_id
    JOIN categorydim              cate
    ON cate.category_id = pc.category_id
WHERE
    category_type = 'meal'
GROUP BY
    category_description,
    the_month
ORDER BY
    category_description,
    the_month;
```

Level 2 Output

	MEAL	MONTH	Monthly Total Sales	CUM_SALES
1	Dessert	01	5,452.54	5,452.54
2	Dessert	02	6,871.67	12,324.20
3	Dessert	03	7,146.57	19,470.77
4	Dessert	04	7,653.11	27,123.88
5	Dessert	05	8,862.61	35,986.49
6	Dessert	06	6,733.36	42,719.85
7	Dessert	07	8,501.60	51,221.45
8	Dessert	08	6,551.78	57,773.22
9	Dessert	09	3,166.07	60,939.29
10	Dessert	10	2,296.14	63,235.43
11	Dessert	11	2,419.78	65,655.21
12	Dessert	12	2,792.12	68,447.33
13	Drink	01	839.14	839.14
14	Drink	02	820.60	1,659.74
15	Drink	03	644.41	2,304.16
16	Drink	04	643.48	2,947.64
17	Drink	05	844.72	3,792.36
18	Drink	06	900.07	4,692.43
19	Drink	07	888.87	5,581.30
20	Drink	08	944.56	6,525.86
21	Drink	09	212.33	6,738.19
22	Drink	10	294.21	7,032.40
23	Drink	11	180.69	7,213.09

Total rows: 48

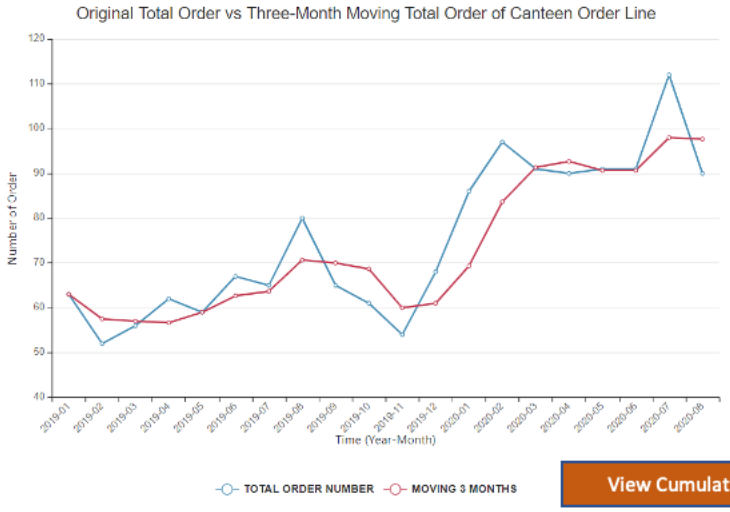
Level 0 OLAP QUERY

```
SELECT
    category_description AS meal,
    to_char(sc_orderdate, 'MM') AS month,
    to_char(SUM(total_sales *(sol_lineprice / sc_totalprice)), '9,999,999,999.99'
    ) AS "Monthly Total Sales",
    to_char(SUM(SUM(total_sales *(sol_lineprice / sc_totalprice))) OVER(
        PARTITION BY category_description
        ORDER BY
            category_description, to_char(sc_orderdate, 'MM')
        ROWS UNBOUNDED PRECEDING
    ), '9,999,999,999.99') AS cum_sales
FROM
    canteenservicefact_0        c
    JOIN customertypedim_0          cust
    ON cust.cust_type_id = c.cust_type_id
    JOIN canteenorderdim_0          co
    ON c.sc_orderid = co.sc_orderid
    JOIN canteenorderlinedim_0      col
    ON co.sc_orderid = col.sc_orderid
    JOIN productdim_0               p
    ON p.product_id = col.product_id
    JOIN productcategorybridgedim_0   pc
    ON p.product_id = pc.product_id
    JOIN categorydim_0              cate
    ON cate.category_id = pc.category_id
WHERE
    category_type = 'meal'
GROUP BY
    category_description,
    to_char(sc_orderdate, 'MM')
ORDER BY
    category_description,
    to_char(sc_orderdate, 'MM');
```

Level 0 Output

	MEAL	MONTH	Monthly Total Sales	CUM_SALES
1	Dessert	01	5,452.54	5,452.54
2	Dessert	02	6,871.67	12,324.20
3	Dessert	03	7,146.57	19,470.77
4	Dessert	04	7,653.11	27,123.88
5	Dessert	05	8,862.61	35,986.49
6	Dessert	06	6,733.36	42,719.85
7	Dessert	07	8,501.60	51,221.45
8	Dessert	08	6,551.78	57,773.22
9	Dessert	09	3,166.07	60,939.29
10	Dessert	10	2,296.14	63,235.43
11	Dessert	11	2,419.78	65,655.21
12	Dessert	12	2,792.12	68,447.33
13	Drink	01	839.14	839.14
14	Drink	02	820.60	1,659.74
15	Drink	03	644.41	2,304.16
16	Drink	04	643.48	2,947.64
17	Drink	05	844.72	3,792.36
18	Drink	06	900.07	4,692.43
19	Drink	07	888.87	5,581.30
20	Drink	08	944.56	6,525.86
21	Drink	09	212.33	6,738.19
22	Drink	10	294.21	7,032.40
23	Drink	11	180.69	7,213.09

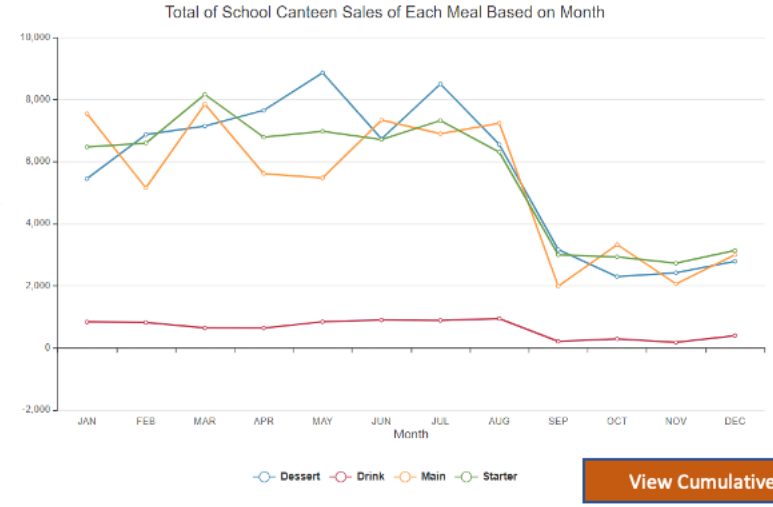
Total rows: 48



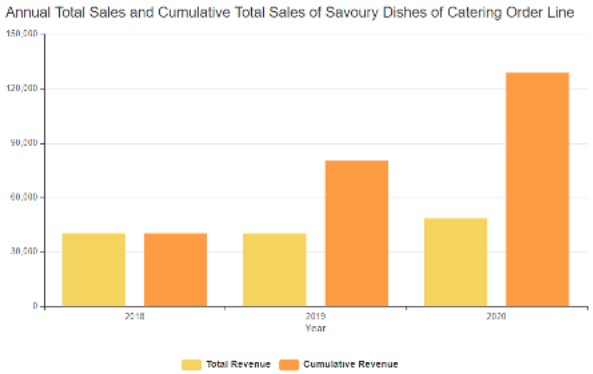
View Cumulative

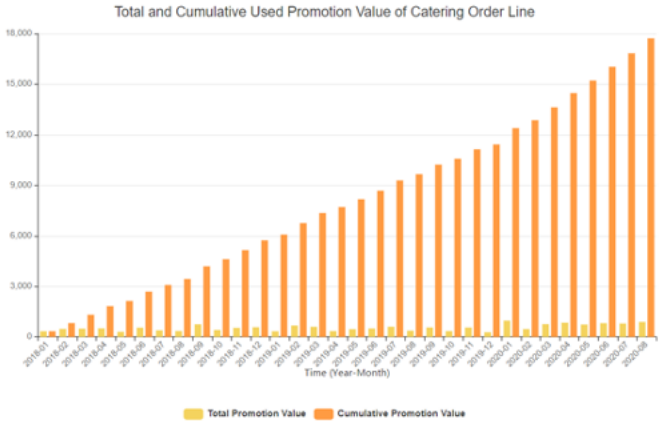
MonChef

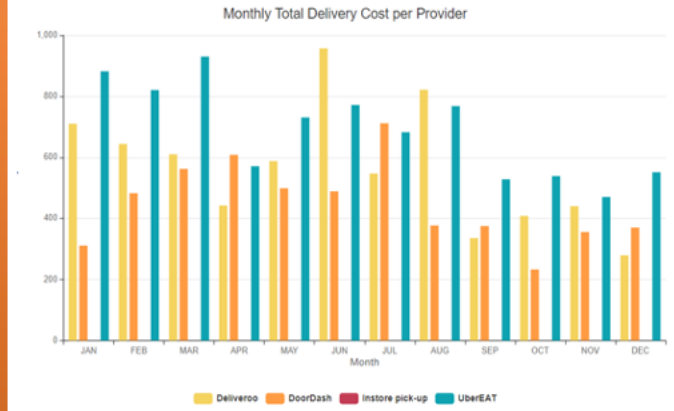
BI Dashboard



View Cumulative







View All Charts

Generate Written Report

Select Year Range

Select Month Range