TIS 3351
ADVANCED DATABASE

Assignment 1

LECTURE SECTION: TC1L

TUTORIAL SECTION: TT1L

BY

NO	STUDENT NAME	STUDENT ID
1	Muhammad Waiee bin Zainol	1191103225
2	Wan Aqel Hakimi Bin Mohd Zamri	1211305491
3	Ilyani Shahnaz binti Shukor	1201101761
4	Tan Hooi Well	1211305474

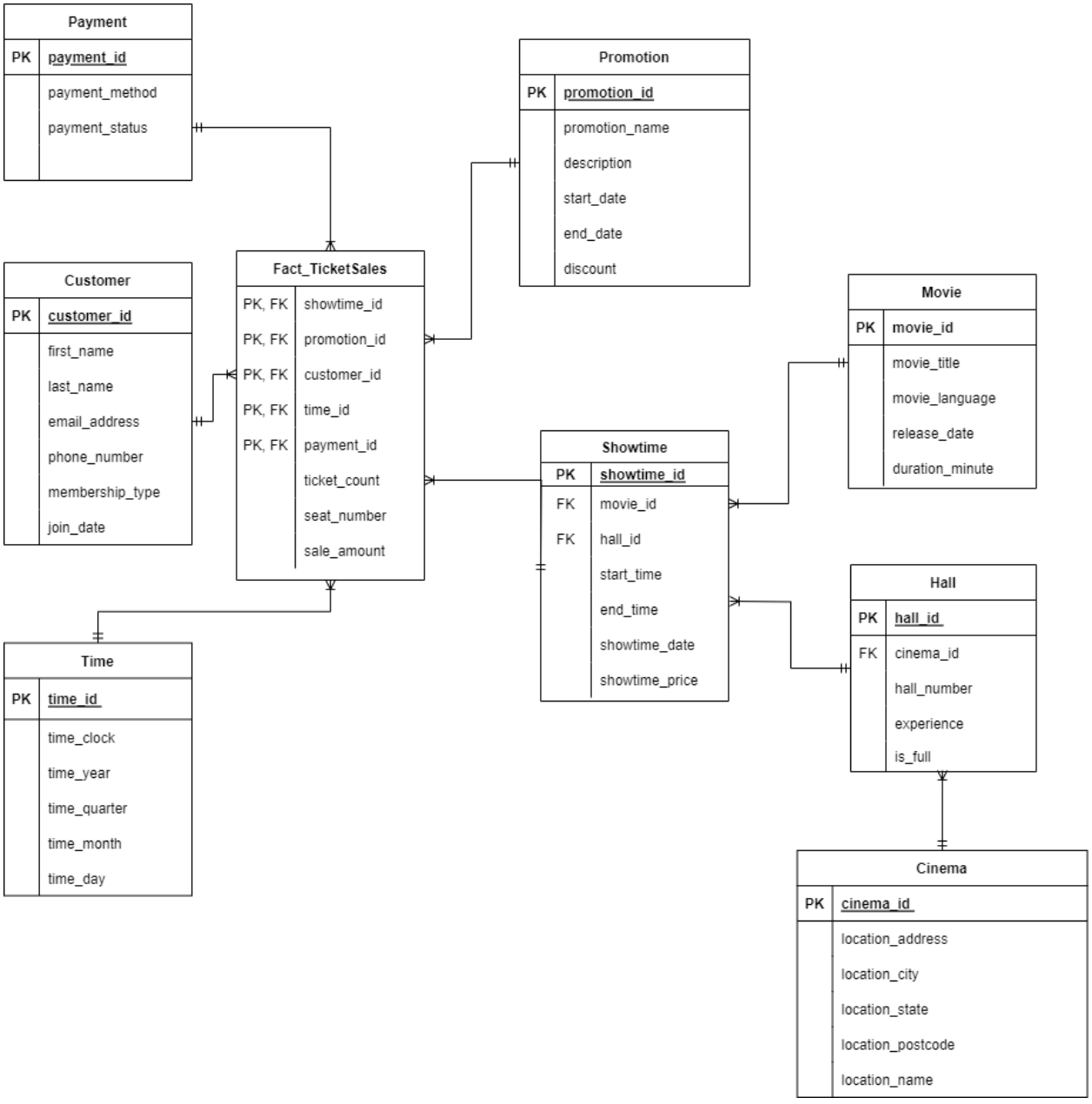
TO

LECTURER NAME	Dr. Haw Su Cheng
GROUP	7

MULTIMEDIA UNIVERSITY
TRIMESTER 2, SESSION 2023/2024

Table of Content

1. Snowflake Schema



2. Data Dictionary

Table Name	Attributes Name	Contents	Data Type	Format	Range	Required	PK/FK	FK Referenced Table
Fact_TicketSales	showtime_id	Id of the showtime	char(10)	ST99999999		Y	PK,FK	showtime_id
	promotion_id	Id of the promotion	char(10)	PRM9999999		Y	PK, FK	promotion_id
	cust_id	Id of the customer	char(10)	CUS9999999		Y	PK, FK	customer_id
	time_id	Id of the booking time	char(10)	TIM9999999		Y	PK, FK	time_id
	payment_id	Id of the payment	char(10)	PAY9999999		Y	PK, FK	payment_id
	ticket_count	Number of tickets bought	int			Y	-	-
	seat_number	Seat number in the hall	varchar(3)	A99		Y	-	-
	sales_amount	Total price of the movie ticket	decimal(5,2)	999.99		Y	-	-
Customer	customer_id	Id of the customer	char(10)	CUS9999999		Y	PK	-
	first_name	Customer's first name	varchar(15)	Xxxxxxx		Y	-	-
	last_name	Customer's last name	varchar(15)	Xxxxxxx		Y	-	-
	email_address	Customer's email address	varchar(50)	Xxxxx@xxxx.xxx		Y	-	-
	phone_number	Customer's phone number	varchar(20)	999-9999-999		Y	-	-
	membership_type	Membership type of the customer	varchar(10)	Xxxxxxx	'	Y	-	-
	join_date	Customer's join date as a member	date	yyyy-mm-dd		-	-	-
Time	time_id	Id of the booking time	char(10)	TIM9999999		Y	PK	-

	time_clock	Time of the reservation	char(10)	Xx:xx	00:00-23:59	Y	-	-
	time_year	Year of the reservation	char(4)	Xxxx		Y	-	-
	time_quarter	Year quarter of the reservation	int		1-4	Y	-	-
	time_month	Month of the reservation	int		1-12	Y	-	-
	time_day	Day of the month of the reservation	int		1-31	Y	-	-
Movie	movie_id	Id of the movie	char(10)	MOV9999999		Y	PK	-
	movie_title	Title of the movie	varchar(50)	Xxxxxxx		Y	-	-
	movie_language	Main language used in the movie	varchar(50)	Xxxxxxx		Y	-	-
	release_date	Release date of the movie	date	yyyy-mm-dd		Y	-	-
	duration_minute	Duration of the movie in minutes	int			Y	-	-
Showtime	showtime_id	Id of the showtime	char(10)	ST99999999		Y	PK	
	movie_id	Id of the movie	char(10)	MOV9999999		Y	FK	movie_id
	hall_id	Id of the hall	char(10)	HLL9999999		Y	FK	hall_id
	start_time	Start time of the movie	char(5)	23:59	00:00-23:59	Y	-	-
	end_time	End time of the movie	char(5)	23:59	00:00-23:59	Y	-	-
	showtime_date	Date of the showtime for the movie	date	yyyy-mm-dd		Y	-	-
	showtime_price	Price of the showtime	decimal(5,2)	999.99		Y	-	-
Hall	hall_id	Id of the hall	char(10)	HLL9999999		Y	PK	-
	cinema_id	Id of the cinema	char(10)	CIN9999999		Y	FK	cinema_id

	hall_number	Number of the hall	int		1-99999	Y	-	-
	experience	Movie experience of the hall	varchar(20)	Xxxxxxx		Y	-	-
	is_full	Determine whether the hall is full or not	char(1)	X		Y	-	-
Cinema	cinema_id	Id of the cinema	char(10)	CIN9999999		Y	PK	-
	location_address	street address of the cinema	varchar(100)	Xxxxxxx		Y	-	-
	location_city	Name city where the cinema located	varchar(50)	Xxxxxxx		Y	-	-
	location_state	Name of state where the cinema located	varchar(50)	Xxxxxxx		Y	-	-
	location_postcode	Postcode of the city where the cinema located	char(5)	99999		Y	-	-
	location_name	Name of the mall where the cinema located	varchar(50)	Xxxxxxx		Y	-	-
Promotion	promotion_id	Id of the promotion	char(10)	PRM9999999		Y	PK	-
	promotion_name	Name of the promotion	varchar(50)	Xxxxxxx		Y	-	-
	description	Description of the promotion	varchar(100)	Xxxxxxx		Y	-	-
	start_date	Date where the promotion starts	date	yyyy-mm-dd		Y	-	-
	end_date	Date where the promotion ends	date	yyyy-mm-dd		Y	-	-
	discount	Discount of the promotion	decimal(4,2)	99.99		Y	-	-
Payment	payment_id	Id of the payment	char(10)	PAY9999999		Y	PK	-
	payment_method	Method of the payment	varchar(20)	Xxxxxxx		Y	-	-
	payment_status	Status of the payment	varchar(20)	Xxxxxxx		Y	-	-

3. Database Schema Fact Table Size and Storage Calculation

There are 5 dimensions in the fact table, each with 10 records. Therefore,

Size of the Fact table (rows) = 10 * 10 * 10 * 10 * 10
= 100,000 rows
= 1 * 10^5 rows

Attributes of Fact Table:

1. showtime_id - char(10)
2. promotion_id - char(10)
3. customer_id - char(10)
4. time_id - char(10)
5. payment_id - char(10)
6. ticket_count - int
7. seat_number - varchar(3)
8. sale_amount - decimal(5,2)

Average bytes per field = (10 + 10 + 10 + 10 + 10 + 4 + 3 + ((5/2)+1))/8
= (57 + 3)/8
= 7.5 bytes

Total storage size for Fact Table = 100,000 rows * 7.5 bytes * 8
= 6,000,000 bytes = 6 MB

4. Data Warehouse Implementation

4.1. Payment Dimension

```
CREATE TABLE Payment
(
      payment_id char(10) NOT NULL PRIMARY KEY,
      payment_method varchar(20) NOT NULL,
      payment_status varchar(20) NOT NULL CHECK (payment_status
IN('Pending','Completed'))
);
```

4.2. Customer Dimension

```
CREATE TABLE Customer
(
      cust_id char(10) NOT NULL PRIMARY KEY,
      first_name varchar(15) NOT NULL,
      last_name varchar(15) NOT NULL,
      email_address varchar(50) NOT NULL,
      phone_number varchar(20) NOT NULL,
      membership_type varchar(10) NOT NULL CHECK (membership_type IN
('Member', 'Non-Member')),
      join_date date
);
```

4.3. Time Dimension

```
CREATE TABLE Time
(
    time_id CHAR(10) NOT NULL PRIMARY KEY,
    time_clock CHAR(5) NOT NULL,
    time_year CHAR(4) NOT NULL,
    time_quarter INT NOT NULL CHECK (time_quarter BETWEEN 1 AND 4),
```

```
    time_month INT NOT NULL CHECK (time_month BETWEEN 1 AND 12),
    time_day INT NOT NULL CHECK (time_day BETWEEN 1 AND 31),
    CHECK (time_quarter = 1 AND time_month BETWEEN 1 AND 3 OR
           time_quarter = 2 AND time_month BETWEEN 4 AND 6 OR
           time_quarter = 3 AND time_month BETWEEN 7 AND 9 OR
           time_quarter = 4 AND time_month BETWEEN 10 AND 12)
);
```

4.4. Promotion Dimension

```
CREATE TABLE Promotion
(
      promotion_id char(10) NOT NULL PRIMARY KEY,
      promotion_name varchar(50) NOT NULL,
      description varchar(100),
      start_date date,
      end_date date,
      discount decimal(4,2)
);
```

4.5. Movie Dimension

```
CREATE TABLE Movie
(
      movie_id char(10) NOT NULL PRIMARY KEY,
      movie_title varchar(50) NOT NULL,
      movie_language varchar(15) NOT NULL,
      release_date date NOT NULL,
      duration_minute int NOT NULL
);
```

4.6. Showtime Dimension

```
CREATE TABLE Showtime
(
	showtime_id char(10) NOT NULL PRIMARY KEY,
	movie_id char(10) NOT NULL,
	hall_id char(10) NOT NULL,
	start_time char(5) NOT NULL,
	end_time char(5) NOT NULL,
	showtime_date date NOT NULL,
	showtime_price decimal(5,2) NOT NULL,
	FOREIGN KEY (movie_id) REFERENCES Movie(movie_id),
	FOREIGN KEY (hall_id) REFERENCES Hall(hall_id)
);
```

4.7. Hall Dimension

```
CREATE TABLE Hall
(
	hall_id char(10) NOT NULL PRIMARY KEY,
	cinema_id char(10) NOT NULL,
	hall_number int NOT NULL,
	experience varchar(20) NOT NULL,
	is_full char(1) NOT NULL,
	FOREIGN KEY (cinema_id) REFERENCES Cinema(cinema_id)
);
```

4.8. Cinema Dimension

```
CREATE TABLE Cinema
(
	cinema_id char(10) NOT NULL PRIMARY KEY,
	location_address varchar(100) NOT NULL,
```

```
      location_city varchar(50) NOT NULL,
      location_state varchar(50) NOT NULL,
      location_postcode char(5) NOT NULL,
      location_name varchar(50) NOT NULL
);
```

4.9. Fact_TicketSales

```
CREATE TABLE Fact_TicketSales
(
         showtime_id CHAR(10) NOT NULL,
         promotion_id CHAR(10) NOT NULL,
         cust_id CHAR(10) NOT NULL,
         time_id CHAR(10) NOT NULL,
         payment_id CHAR(10) NOT NULL,
         ticket_count INT NOT NULL,
         seat_number VARCHAR(3) NOT NULL,
         sale_amount DECIMAL(5,2) NOT NULL DEFAULT 00.00,
         PRIMARY KEY (showtime_id, promotion_id, cust_id, time_id,
payment_id),
         FOREIGN KEY (showtime_id) REFERENCES Showtime(showtime_id),
         FOREIGN KEY (promotion_id) REFERENCES Promotion(promotion_id),
            FOREIGN KEY (cust_id) REFERENCES Customer(cust_id),
         FOREIGN KEY (time_id) REFERENCES Time(time_id),
         FOREIGN KEY (payment_id) REFERENCES Payment(payment_id)
);
```

5. Sample Data

5.1. Payment Dimension

```
INSERT INTO Payment (payment_id, payment_method, payment_status)
VALUES
```

```
    ('PAY0000001', 'Credit Card', 'Completed'),
    ('PAY0000002', 'Debit Card', 'Completed'),
    ('PAY0000003', 'Gift Card', 'Completed'),
    ('PAY0000004', 'E-Wallet', 'Pending'),
    ('PAY0000005', 'Credit Card', 'Completed'),
    ('PAY0000006', 'Debit Card', 'Pending'),
    ('PAY0000007', 'E-Wallet', 'Completed'),
    ('PAY0000008', 'Debit Card', 'Completed'),
    ('PAY0000009', 'Debit Card', 'Pending'),
    ('PAY0100000', 'Credit Card', 'Completed')
```

5.2. Customer Dimension

```
INSERT INTO Customer (cust_id, first_name, last_name, email_address,
phone_number, membership_type, join_date)
VALUES
    ('CUS0000001', 'Alice', 'Anderson', 'alice@gmail.com', '012-456-7890',
'Non-Member', '2023-01-01'),
    ('CUS0000002', 'Bob', 'Brown', 'bob@gmail.com','010-654-3210', 'Member',
'2023-01-02'),
    ('CUS0000003', 'Charlie', 'Clark', 'charlie@gmail.com', '011-789-0123',
'Member', '2023-01-03'),
    ('CUS0000004', 'David', 'Davis', 'david@gmail.com', '018-012-3456',
'Member', '2023-01-04'),
    ('CUS0000005', 'Eva', 'Evans', 'eva@gmail.com', '018-345-6789',
'Non-Member', '2023-01-05'),
    ('CUS0000006', 'Frank', 'Franklin', 'frank@gmail.com', '011-678-9012',
'Member', '2023-01-06'),
    ('CUS0000007', 'Grace', 'Gray', 'grace@gmail.com', '014-901-2345',
'Member', '2023-01-07'),
    ('CUS0000008', 'Harry', 'Harrison', 'harry@gmail.com', '019-234-5678',
'Member', '2023-01-08'),
    ('CUS0000009', 'Ivy', 'Irwin', 'ivy@gmail.com', '019-567-8901',
```

```
'Non-Member', '2023-01-09'),
    ('CUS0000010', 'Jack', 'Jackson', 'jack@gmail.com', '010-890-1234',
'Member' , '2023-01-10')
```

5.3. Time Dimension

```
INSERT INTO Time (time_id, time_clock, time_year, time_quarter, time_month,
time_day)
VALUES
    ('TIM0000001', '09:00', '2023' , '4' , '12', '01'),
    ('TIM0000002', '10:30', '2023' , '4' , '12', '02'),
    ('TIM0000003', '11:00', '2023' , '4' , '12', '03'),
    ('TIM0000004', '15:30', '2023' , '4' , '12', '04'),
    ('TIM0000005', '14:30', '2023' , '4' , '12', '05'),
    ('TIM0000006', '15:00', '2023' , '4' , '12', '06'),
    ('TIM0000007', '10:15', '2023' , '4' , '12', '07'),
    ('TIM0000008', '14:40', '2023' , '4' , '12', '08'),
    ('TIM0000009', '08:10', '2023' , '4' , '12', '09'),
    ('TIM0000010', '19:00', '2023' , '4' , '12', '10')
```

5.4. Promotion Dimension

```
INSERT INTO Promotion (promotion_id, promotion_name, description, start_date,
end_date, discount)
VALUES
    ('PRM0000000','NoPromotion', NULL,NULL,NULL, 0.00),
    ('PRM0000001', 'CNY', 'Chinese New Year!', '2023-03-01', '2023-03-31',
0.15),
    ('PRM0000002', 'BlackFriday', 'Black Friday', '2023-04-01', '2023-04-30',
0.20),
    ('PRM0000003', 'BackToSchool', 'Special discounts on back-to-school
essentials.', '2023-05-01', '2023-05-30', 0.10),
```

```
    ('PRM0000004', 'FallFest', 'Celebrate fall with exclusive discounts',
'2023-06-15', '2023-07-15', 0.18),
    ('PRM0000005', 'SpringSale', 'Enjoy spring with special discounts!',
'2023-08-01', '2023-08-30', 0.12),
    ('PRM0000006', 'SummerSplash', 'Cool off with summer savings!',
'2023-09-01', '2023-09-30', 0.15),
    ('PRM0000007', 'HolidayJoy', 'Spread holiday cheer with exclusive
promotions.', '2023-10-01', '2023-10-31', 0.25),
    ('PRM0000008', 'TechFrenzy', 'Unleash the tech enthusiast in you with
amazing tech deals.', '2023-11-01', '2023-11-30', 0.18),
    ('PRM0000009', 'WinterWarmth', 'Stay cozy this winter with special
discounts on winter essentials.', '2023-12-01', '2023-12-31', 0.20);
```

5.5. Movie Dimension

```
INSERT INTO Movie (movie_id,  movie_title, movie_language, release_date,
duration_minute) VALUES
    ('MOV0000001', 'War On Terror : KL Anarki', 'Malay', '2023-11-23', 100),
    ('MOV0000002', 'Wakaf', 'Indo', '2023-12-07', 99),
    ('MOV0000003', 'Wish', 'English', '2023-11-23', 94),
    ('MOV0000004', 'Trending Topic', 'Mandarin', '2023-12-7', 122),
    ('MOV0000005', 'Magik', 'Malay', '2023-11-30', 113),
    ('MOV0000006', 'Hi Nanna', 'Tamil', '2023-12-7', 142),
    ('MOV0000007', 'Cobweb', 'Korean', '2023-11-16', 133),
    ('MOV0000008', 'Silent Night', 'English', '2023-11-30', 104),
    ('MOV0000009', 'Gampang Cuan', 'Indo', '2023-12-07', 119),
    ('MOV0000010', 'The Marvels', 'English', '2023-11-09', 105)
```

5.6. Showtime Dimension

```
INSERT INTO Showtime (showtime_id, movie_id, hall_id, start_time, end_time,
```

```
showtime_date, showtime_price)
VALUES
    ('ST0000001', 'MOV0000001', 'HLL0000001', '14:00', '16:28', '2023-02-01',
24.00),
    ('ST0000002', 'MOV0000003', 'HLL0000002', '18:30', '20:42', '2023-03-01',
24.00),
    ('ST0000003', 'MOV0000004', 'HLL0000002', '12:15', '14:20', '2023-04-01',
18.00),
    ('ST0000004', 'MOV0000002', 'HLL0000001', '16:45', '18:53', '2023-05-04',
15.00),
    ('ST0000005', 'MOV0000005', 'HLL0000003', '15:45', '18:18', '2023-06-15',
20.00),
    ('ST0000006', 'MOV0000007', 'HLL0000004', '20:00', '22:13', '2023-08-06',
22.00),
    ('ST0000007', 'MOV0000009', 'HLL0000005', '14:30', '16:49', '2023-09-07',
18.00),
    ('ST0000008', 'MOV0000010', 'HLL0000006', '17:15', '19:30', '2023-10-08',
25.00),
    ('ST0000009', 'MOV0000006', 'HLL0000007', '12:45', '15:17', '2023-11-09',
18.50),
    ('ST0000010', 'MOV0000008', 'HLL0000008', '19:30', '21:34', '2023-12-12',
23.50);
```

5.7. Hall Dimension

```
INSERT INTO Hall (hall_id, cinema_id, hall_number, experience, is_full)
VALUES
    ('HLL0000001', 'CIN0000001',  5, 'Standard', 0),
    ('HLL0000002', 'CIN0000002',  9, 'IMAX', 0),
    ('HLL0000003', 'CIN0000003',  1, 'Standard', 0),
    ('HLL0000004', 'CIN0000004',  3, 'Dolby Atmos', 0),
```

```
    ('HLL0000005', 'CIN0000005',  6, '4DX', 0),
    ('HLL0000006', 'CIN0000006',  2, 'Standard', 0),
    ('HLL0000007', 'CIN0000007',  4, 'IMAX', 0),
    ('HLL0000008', 'CIN0000008',  8, 'Dolby Atmos', 0),
    ('HLL0000009', 'CIN0000009',  7, 'Standard', 0),
    ('HLL0000010', 'CIN0000010',  10, '4DX', 0);
```

5.8. Cinema Dimension

```
INSERT INTO Cinema (cinema_id, location_address, location_city,
location_state, location_postcode, location_name) VALUES
    ('CIN0000001', 'L3-AT5, 2nd Floor, IOI City Mall', 'Putrajaya',
'Selangor', '62502', 'GSC IOI City Mall'),
    ('CIN0000002', '3RD FLOOR, Lot T-001 Mid Valley Megamall, Lingkaran Syed
Putra, Mid Valley City', 'Kuala Lumpur', 'Federal Territory of Kuala Lumpur',
'59200', 'GSC Cinema Mid Valley Megamall'),
    ('CIN0000003', 'Lot L5.14, Level 5 Nu Sentral, 201, Jalan Tun Sambanthan',
'Kuala Lumpur', 'Federal Territory of Kuala Lumpur', '50470', 'GSC NU
Sentral'),
    ('CIN0000004', 'Lot S 29A & S, 30, Floor 2, Jln Taman Ibu Kota, Danau
Kota', 'Kuala Lumpur', 'Federal Territory of Kuala Lumpur', '53300', 'GSC
Setapak Central'),
    ('CIN0000005', 'Lot F30, 32 & 33A, Subang Parade, 5, Jalan SS 16/1, Ss
16', 'Subang Jaya', 'Selangor', '47500', 'GSC Subang Parade'),
    ('CIN0000006', 'T-01, Level 2A, EkoCheras Mall No 693, Batu, 5, Jln
Cheras', 'Kuala Lumpur', 'Federal Territory of Kuala Lumpur', '56000', 'GSC
EkoCheras Mall'),
    ('CIN0000007', 'MyTOWN Shopping Centre, Level 3A & 3B, Seksyen 90,
L3-AT-002, Jalan Cochrane, Maluri', 'Kuala Lumpur', 'Federal Territory of
Kuala Lumpur', '55100', 'GSC MyTown'),
    ('CIN0000008', 'G3-18, Level G3, Jln Lingkaran Tengah 2, KL Timur', 'Kuala
Lumpur', 'Federal Territory of Kuala Lumpur', '53100', 'GSC KL East Mall'),
    ('CIN0000009', 'The Summit Subang, Level 3 & 5, Persiaran Kewajipan, Usj
```

```
1', 'Subang Jaya', 'Selangor', '47600', 'GSC Summit USJ'),
    ('CIN0000010', '3rd Floor, Tropicana Gardens Mall, No2A, Persiaran Surian,
Tropicana Indah', 'Petaling Jaya', 'Selangor', '47810', 'GSC Tropicana Gardens
Mall');
```

5.9. Fact_TicketSales

```
INSERT INTO Fact_TicketSales (showtime_id, promotion_id, cust_id, time_id,
payment_id, ticket_count, seat_number, sale_amount)
VALUES
    ('ST0000001', 'PRM0000000', 'CUS0000001', 'TIM0000001', 'PAY0000001', 1,
'G12', DEFAULT),
    ('ST0000002', 'PRM0000001', 'CUS0000002', 'TIM0000002', 'PAY0000002', 3,
'E09', DEFAULT),
    ('ST0000003', 'PRM0000002', 'CUS0000003', 'TIM0000003', 'PAY0000003', 4,
'I04', DEFAULT),
    ('ST0000004', 'PRM0000003', 'CUS0000004', 'TIM0000004', 'PAY0000003', 2,
'F02', DEFAULT),
    ('ST0000005', 'PRM0000004', 'CUS0000010', 'TIM0000005', 'PAY0000004', 2,
'A03', DEFAULT),
    ('ST0000006', 'PRM0000005', 'CUS0000005', 'TIM0000006', 'PAY0000005', 3,
'B08', DEFAULT),
    ('ST0000007', 'PRM0000006', 'CUS0000006', 'TIM0000007', 'PAY0000006', 1,
'C12', DEFAULT),
    ('ST0000008', 'PRM0000007', 'CUS0000007', 'TIM0000008', 'PAY0000007', 4,
'D05', DEFAULT),
    ('ST0000009', 'PRM0000008', 'CUS0000008', 'TIM0000009', 'PAY0000008', 2,
'E09', DEFAULT),
    ('ST0000010', 'PRM0000009', 'CUS0000009', 'TIM0000010', 'PAY0000009', 3,
'F14', DEFAULT);
```

6. Procedural SQLs

6.1. Stored procedure

The stored procedure "UpdateMembershipStatus" is designed to automatically update the membership status of a customer in the Customer table based on their total purchase amount, utilizing information from the Fact_TicketSales table. The procedure takes a customer ID, "p_cust_id," as input and initializes variables to store the total purchase amount, "v_total_purchase," and the new membership status, "v_new_membership_status." It then calculates the total purchase amount by summing the sale amounts from the Fact_TicketSales table for the specified customer ID. Subsequently, the procedure determines the new membership status by evaluating whether the total purchase amount is equal to or exceeds 500. If so, the customer is set as a 'Member'; otherwise, they are categorized as a 'Non-Member.' Finally, the Customer table is updated with the new membership status for the provided customer ID.

```
CREATE PROCEDURE UpdateMembershipStatus(
   IN p_cust_id CHAR(10)
)
BEGIN
   DECLARE v_total_purchase DECIMAL(10, 2) DEFAULT 0;
   DECLARE v_new_membership_status VARCHAR(10);

   -- Calculate total purchase amount for the customer
   SELECT SUM(ts.sale_amount) INTO v_total_purchase

   FROM Fact_TicketSales ts
   WHERE ts.cust_id = p_cust_id;
   IF v_total_purchase >= 500 THEN
       SET v_new_membership_status = 'Member';
   ELSE
       SET v_new_membership_status = 'Non-Member';
   END IF;
```

```
   -- Update the Customer table with the new membership status
   UPDATE Customer
   SET membership_type = v_new_membership_status
   WHERE cust_id = p_cust_id;
END
```

To validate this stored procedure, values of the sale_amount for customer_id 'CUS0000001' are set to 520 to meet the requirement above 500, while 'CUS0000002' are set to 200 indicating that it is below 500 in the Fact_TiketSales. While in the Customer table, their membership_type is set to 'Non-Member'.

Before Stored Procedure:

Fact Table

SHOWTIME_ID	PROMOTION_ID	CUST_ID	TIME_ID	PAYMENT_ID	TICKET_COUNT	SEAT_NUMBER	SALE_AMOUNT
ST0000001	PRM0000000	CUS0000001	TIM0000001	PAY0000001	1	W09	520
ST0000002	PRM0000001	CUS0000002	TIM0000002	PAY0000002	3	E09	200

Customer Table

CUST_ID	FIRST_NAME	LAST_NAME	EMAIL_ADDRESS	PHONE_NUMBER	MEMBERSHIP_TYPE	JOIN_DATE
CUS0000001	Alice	Anderson	alice@gmail.com	012-456-7890	Non-Member	2023-01-01
CUS0000002	Bob	Brown	bob@gmail.com	010-654-3210	Non-Member	2023-01-02

To execute the stored procedure, the call statement is inserted as follows:

```
CALL UpdateMembershipStatus('CUS0000001');

CALL UpdateMembershipStatus('CUS0000002');
```

After Stored Procedure:

Customer Table

CUST_ID	FIRST_NAME	LAST_NAME	EMAIL_ADDRESS	PHONE_NUMBER	MEMBERSHIP_TYPE	JOIN_DATE
CUS0000001	Alice	Anderson	alice@gmail.com	012-456-7890	Member	2023-01-01
CUS0000002	Bob	Brown	bob@gmail.com	010-654-3210	Non-Member	2023-01-02

Notice that customer ID ‘CUS0000001’ has become a ‘Member’ since the customer sale amount is above 500 in cinema while ‘CUS0000001’ remains ‘Non-Member’ since the total sale amount of the customer did not exceed 500 yet.

6.2. Trigger

The trigger `calculate_sale_amount` aims to calculate the sale_amount in the Fact_TicketSales table and is executed after an insert statement into Fact_TicketSales. For rows where the payment status from the Payment table is 'Completed', the sale_amount is calculated by multiplying the showtime_price (retrieved from the Showtime table), the ticket_count from the Fact_TicketSales table, and a discount (retrieved from the Promotion table). On the other hand, if the payment status is 'Pending', the sale_amount is set to 0.00, as the transaction has yet to be completed.

```
CREATE TRIGGER calculate_sale_amount
AFTER INSERT ON Fact_TicketSales
FOR EACH ROW MODE DB2SQL

BEGIN
    -- Update the sale_amount in the Fact_TicketSales table
    UPDATE Fact_TicketSales ft
    SET sale_amount =
        CASE
        --If payment_status = 'Completed', calculate the showtime price
            WHEN (SELECT payment_status
                  FROM Payment py
                  WHERE py.payment_id = ft.payment_id) = 'Completed'
            THEN (SELECT showtime_price
                  FROM Showtime s
                  WHERE s.showtime_id = ft.showtime_id) * ft.ticket_count *
                 (1 - (SELECT discount
                       FROM Promotion p
                       WHERE p.promotion_id = ft.promotion_id))
          --If payment_status = 'Pending', set as 0
            WHEN (SELECT payment_status
                  FROM Payment py
                  WHERE py.payment_id = ft.payment_id) = 'Pending'
```

```
            THEN 0.00
        END;
END;
```

Before the execution of the trigger, all values of the sale_amount are set to the default value of 0 as specified in the previous table creation and value insertion.

Before Trigger:

	SHOWTIME_ID	PROMOTION_ID	CUST_ID	TIME_ID	PAYMENT_ID	TICKET_COUNT	SEAT_NUMBER	SALE_AMOUNT
1	ST0000001	PRM0000000	CUS0000001	TIM0000001	PAY0000001	1	G12	0
2	ST0000002	PRM0000001	CUS0000002	TIM0000002	PAY0000002	3	E09	0
3	ST0000003	PRM0000002	CUS0000003	TIM0000003	PAY0000003	4	I04	0
4	ST0000004	PRM0000003	CUS0000004	TIM0000004	PAY0000003	2	F02	0
5	ST0000005	PRM0000004	CUS0000010	TIM0000005	PAY0000004	2	A03	0
6	ST0000006	PRM0000005	CUS0000005	TIM0000006	PAY0000005	3	B08	0
7	ST0000007	PRM0000006	CUS0000006	TIM0000007	PAY0000006	1	C12	0
8	ST0000008	PRM0000007	CUS0000007	TIM0000008	PAY0000007	4	D05	0
9	ST0000009	PRM0000008	CUS0000008	TIM0000009	PAY0000008	2	E09	0
10	ST0000010	PRM0000009	CUS0000009	TIM0000010	PAY0000009	3	F14	0

To execute the trigger, the insert statement is inserted as follows:

```
INSERT INTO Fact_TicketSales (showtime_id, promotion_id, cust_id, time_id,
payment_id, ticket_count, seat_number, sale_amount)
VALUES
    ('ST0000005', 'PRM0000009', 'CUS0000002', 'TIM0000001', 'PAY0000005', 1,
'I04', DEFAULT)
```

After Trigger

	SHOWTIME_ID	PROMOTION_ID	CUST_ID	TIME_ID	PAYMENT_ID	TICKET_COUNT	SEAT_NUMBER	SALE_AMOUNT
1	ST0000001	PRM0000000	CUS0000001	TIM0000001	PAY0000001	1	G12	24
2	ST0000002	PRM0000001	CUS0000002	TIM0000002	PAY0000002	3	E09	61.2
3	ST0000003	PRM0000002	CUS0000003	TIM0000003	PAY0000003	4	I04	57.6
4	ST0000004	PRM0000003	CUS0000004	TIM0000004	PAY0000003	2	F02	27
5	ST0000005	PRM0000004	CUS0000010	TIM0000005	PAY0000004	2	A03	0
6	ST0000006	PRM0000005	CUS0000005	TIM0000006	PAY0000005	3	B08	58.08
7	ST0000007	PRM0000006	CUS0000006	TIM0000007	PAY0000006	1	C12	0
8	ST0000008	PRM0000007	CUS0000007	TIM0000008	PAY0000007	4	D05	75
9	ST0000009	PRM0000008	CUS0000008	TIM0000009	PAY0000008	2	E09	30.34
10	ST0000010	PRM0000009	CUS0000009	TIM0000010	PAY0000009	3	F14	0
11	ST0000005	PRM0000009	CUS0000002	TIM0000001	PAY0000005	1	I04	16

To validate the inserted value, the values are retrieved from these tables:

Showtime Table

	SHOWTIME_ID	MOVIE_ID	HALL_ID	START_TIME	END_TIME	SHOWTIME_DATE	SHOWTIME_PRICE
1	ST0000005	MOV0000005	HLL0000003	15:45	18:18	2023-06-15	20

Promotion Table

	PROMOTION_ID	PROMOTION_NAME	DESCRIPTION	START_DATE	END_DATE	DISCOUNT
1	PRM0000009	WinterWarmth	Stay cozy this winter with special discounts on winter essentials.	2023-12-01	2023-12-31	0.2

Payment Table

	PAYMENT_ID	PAYMENT_METHOD	PAYMENT_STATUS
1	PAY0000005	Credit Card	Completed

Since the payment status is completed, the sale amount is calculated by the

showtime_price * ticket_count * 1-discount

20 * 1 * 0.8 = 16

The same value is observed in the fact table in the last row.

6.3. User-defined function

The query aims to retrieve comprehensive information about movie ticket sales for a specific customer identified by the Customer ID '. This information includes details about each ticket purchase, encompassing aspects such as the showtime, movie title, language, start and end times, hall number, cinema location, promotion details, ticket count, showtime price, promotion discount, and the total sales amount for each transaction.

```
CREATE FUNCTION ReturnMovieInfo (CID CHAR(10))
RETURNS TABLE
(
    showtime_id CHAR(10),
    movie_title VARCHAR(50),
    movie_language VARCHAR(15),
    start_time CHAR(5),
    end_time CHAR(5),
    hall_number INT,
    location_name VARCHAR(50),
    promotion_name VARCHAR(50),
    ticket_count INT,
    showtime_price DECIMAL(5,2),
    promotion_discount DECIMAL(3,2),
    total_sales_amount DECIMAL(10,2)
)
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
DETERMINISTIC
RETURN
    SELECT s.showtime_id,
           m.movie_title,
           m.movie_language,
```

```
            s.start_time,
            s.end_time,
            h.hall_number,
            c.location_name,
            p.promotion_name,
            f.ticket_count,
            s.showtime_price,
            p.discount AS promotion_discount,
            (s.showtime_price * f.ticket_count - (s.showtime_price *
f.ticket_count * p.discount)) AS total_sales_amount
    FROM Fact_TicketSales f
    JOIN Showtime s ON f.showtime_id = s.showtime_id
    JOIN Hall h ON s.hall_id = h.hall_id
    JOIN Cinema c ON h.cinema_id = c.cinema_id
    JOIN Movie m ON s.movie_id = m.movie_id
    JOIN Promotion p ON f.promotion_id = p.promotion_id
    WHERE f.cust_id = CID;

 SELECT * FROM TABLE (RETURNMOVIEINFO('CUS0000005'))
```

After creating the user defined function, to gain information about customer id = 'CUS0000005', Execute the select statement SELECT * FROM TABLE (RETURNMOVIEINFO('CUS0000005')) and it will show all the information that we select in the function.

	SHOWTIME_ID	MOVIE_TITLE	MOVIE_LANGUAGE	START_TIME	END_TIME	HALL_NUMBER	LOCATION_NAME
1	ST0000006	Cobweb	Korean	20:00	22:13	3	GSC Setapak Central

LOCATION_NAME	PROMOTION_NAME	TICKET_COUNT	SHOWTIME_PRICE	PROMOTION_DISCOUNT	TOTAL_SALES_AMOUNT
GSC Setapak Central	SpringSale	3	22	0.12	58.08

7. Complex Query

7.1. Complex query with joins of at least 3 tables

This query extracts detailed information on completed movie ticket sales during the first half of 2023, shedding light on key aspects of customer transactions, promotions, and cinema operations. The selected columns encompass unique identifiers, transaction details, showtime specifics, promotional attributes, customer demographics, movie details, hall characteristics, and cinema location. By focusing on completed payments, the query ensures the inclusion of successfully concluded transactions, while restricting the results to the specified time frame provides a snapshot of sales trends from January 1 to June 30, 2023. The ordering of results by showtime date and sale amount offers a chronological and financial perspective, aiding in the identification of recent successful transactions.

```
SELECT
    FTS.showtime_id,
    FTS.promotion_id,
    FTS.cust_id,
    FTS.time_id,
    FTS.payment_id,
    FTS.ticket_count,
    FTS.seat_number,
    FTS.sale_amount,
    ST.start_time,
    ST.end_time,
    ST.showtime_date,
    P.promotion_name,
    P.start_date AS promotion_start_date,
    P.end_date AS promotion_end_date,
    C.first_name,
    C.last_name,
    C.email_address,
    C.phone_number,
```

```
    M.movie_title,
    H.hall_number,
    H.experience,
    CI.location_name AS cinema_location,
    CI.location_city AS cinema_city
FROM
    Fact_TicketSales FTS
JOIN
    Showtime ST ON FTS.showtime_id = ST.showtime_id
JOIN
    Promotion P ON FTS.promotion_id = P.promotion_id
JOIN
    Customer C ON FTS.cust_id = C.cust_id
JOIN
    Time T ON FTS.time_id = T.time_id
JOIN
    Payment PM ON FTS.payment_id = PM.payment_id
JOIN
    Movie M ON ST.movie_id = M.movie_id
JOIN
    Hall H ON ST.hall_id = H.hall_id
JOIN
    Cinema CI ON H.cinema_id = CI.cinema_id
WHERE
    T.time_year = '2023'
    AND PM.payment_status = 'Completed'
    AND ST.showtime_date BETWEEN '2023-01-01' AND '2023-06-30'
ORDER BY
    ST.showtime_date DESC, FTS.sale_amount DESC;
```

SALE_AMOUNT	START_TIME	END_TIME	SHOWTIME_DATE	PROMOTION_NAME	PROMOTION_START_DATE	PROMOTION_END_DATE
0	16:45	18:53	2023-05-04	BackToSchool	2023-05-01	2023-05-30
0	12:15	14:20	2023-04-01	BlackFriday	2023-04-01	2023-04-30
0	18:30	20:42	2023-03-01	CNY	2023-03-01	2023-03-31
0	14:00	16:28	2023-02-01	NoPromotion	[NULL]	[NULL]

	SHOWTIME_ID	PROMOTION_ID	CUST_ID	TIME_ID	PAYMENT_ID	TICKET_COUNT	SEAT_NUMBER
1	ST0000004	PRM0000003	CUS0000004	TIM0000004	PAY0000003	2	F02
2	ST0000003	PRM0000002	CUS0000003	TIM0000003	PAY0000003	4	I04
3	ST0000002	PRM0000001	CUS0000002	TIM0000002	PAY0000002	3	E09
4	ST0000001	PRM0000000	CUS0000001	TIM0000001	PAY0000001	1	G12

7.2. Group by Rollup and having clause

The SELECT statement below retrieves and aggregates ticket sales data from the Customer, Movie, Showtime, and Fact_TicketSales tables. It calculates the total tickets and sales amount for each customer and movie combination. The results are grouped using the ROLLUP function to provide subtotals and a grand total. The HAVING clause filters the results to include only those with a total sale amount greater than or equal to $15. The final result set is ordered alphabetically by customer names and movie titles.

```
SELECT
    c.first_name || ' ' || c.last_name AS Full_Name,
    m.movie_title,
    SUM(ft.ticket_count) AS total_tickets,
    SUM(ft.sale_amount) AS Total_Sale
FROM
    Customer c, Movie m, Showtime s, Fact_TicketSales ft
WHERE
    m.movie_id = s.movie_id
    AND s.showtime_id = ft.showtime_id
    AND c.cust_id = ft.cust_id
GROUP BY
    ROLLUP (c.first_name, c.last_name, m.movie_title)
HAVING
    SUM(ft.sale_amount) >= 15
ORDER BY c.first_name, c.last_name, m.movie_title;
```

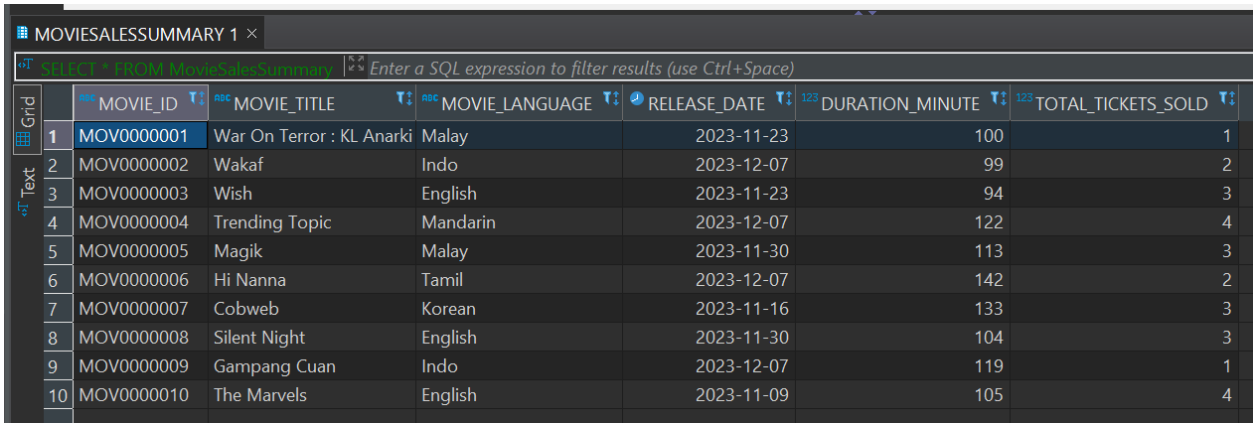
The result of this statement is observed as shown below:

	FULL_NAME	MOVIE_TITLE	TOTAL_TICKETS	TOTAL_SALE
1	Alice Anderson	War On Terror : KL Anarki	1	24
2	Alice Anderson	[NULL]	1	24
3	[NULL]	[NULL]	1	24
4	Bob Brown	Magik	1	16
5	Bob Brown	Wish	3	61.2
6	Bob Brown	[NULL]	4	77.2
7	[NULL]	[NULL]	4	77.2
8	Charlie Clark	Trending Topic	4	57.6
9	Charlie Clark	[NULL]	4	57.6
10	[NULL]	[NULL]	4	57.6
11	David Davis	Wakaf	2	27
12	David Davis	[NULL]	2	27
13	[NULL]	[NULL]	2	27
14	Eva Evans	Cobweb	3	58.08
15	Eva Evans	[NULL]	3	58.08
16	[NULL]	[NULL]	3	58.08
17	Grace Gray	The Marvels	4	75
18	Grace Gray	[NULL]	4	75
19	[NULL]	[NULL]	4	75
20	Harry Harrison	Hi Nanna	2	30.34
21	Harry Harrison	[NULL]	2	30.34
22	[NULL]	[NULL]	2	30.34
23	[NULL]	[NULL]	26	349.22

7.3. View

```
CREATE VIEW MovieSalesSummary AS
SELECT
    m.movie_id,
    m.movie_title,
    m.movie_language,
    m.release_date,
    m.duration_minute,
    SUM(f.ticket_count) AS total_tickets_sold
FROM
    Movie m
JOIN Showtime s ON m.movie_id = s.movie_id
JOIN Fact_TicketSales f ON s.showtime_id = f.showtime_id
GROUP BY
    m.movie_id, m.movie_title, m.movie_language, m.release_date,
m.duration_minute;
```

The view selects specific columns such as movie ID, title, language, release date, duration, and total tickets sold. It achieves this by joining the Movie, Showtime, and Fact_TicketSales tables, and groupubg them by movie ID, title, language, release date and duration.


MOVIESALESSUMMARY 1

SELECT * FROM MovieSalesSummary | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text

	MOVIE_ID	MOVIE_TITLE	MOVIE_LANGUAGE	RELEASE_DATE	DURATION_MINUTE	TOTAL_TICKETS_SOLD
1	MOV0000001	War On Terror : KL Anarki	Malay	2023-11-23	100	1
2	MOV0000002	Wakaf	Indo	2023-12-07	99	2
3	MOV0000003	Wish	English	2023-11-23	94	3
4	MOV0000004	Trending Topic	Mandarin	2023-12-07	122	4
5	MOV0000005	Magik	Malay	2023-11-30	113	3
6	MOV0000006	Hi Nanna	Tamil	2023-12-07	142	2
7	MOV0000007	Cobweb	Korean	2023-11-16	133	3
8	MOV0000008	Silent Night	English	2023-11-30	104	3
9	MOV0000009	Gampang Cuan	Indo	2023-12-07	119	1
10	MOV0000010	The Marvels	English	2023-11-09	105	4

7.4. TWO SQL not covered in lecture

1. DECLARE

The DECLARE keyword is used to define variables within a stored procedure or block of code. In the given example, two variables, "v_total_purchase" and "v_new_membership_status," are declared. These variables serve as placeholders for storing a decimal value and a string, respectively, facilitating data manipulation and storage within the context of the stored procedure used in this data warehouse implementation.

```
-- Variable to store total purchase amount for the customer
DECLARE v_total_purchase DECIMAL(10, 2) DEFAULT 0;
-- Variable for new membership status
DECLARE v_new_membership_status VARCHAR(10);
```

2. CURSOR

A cursor is a mechanism used to traverse and manipulate the result set of a query, typically within a stored procedure or a batch of SQL statements. It provides a way to iterate over a set of rows returned by a SELECT statement, enabling row-level operations.

Below is an example of a cursor used in a stored procedure. In this context, a cursor named "ticket_cursor" is declared to traverse the result set obtained from the "Fact_TicketSales" table. The cursor is opened, and rows are fetched one at a time using the FETCH statement. Inside the WHILE loop, each set of fetched values is printed using the PRINT statement. The loop continues until there are no more rows to fetch. Finally, the cursor is closed and deallocated, completing the row-wise processing of the result set from "Fact_TicketSales." Cursors are useful for iterating through query results and performing specific actions on each row, as demonstrated in this code.

```
DECLARE ticket_cursor CURSOR FOR
```

```
    SELECT showtime_id, promotion_id, cust_id, ticket_count, sale_amount
    FROM Fact_TicketSales;

DECLARE @showtime_id CHAR(10), @promotion_id CHAR(10), @cust_id CHAR(10),
@ticket_count INT, @sale_amount DECIMAL(5, 2);

OPEN ticket_cursor;

FETCH NEXT FROM ticket_cursor INTO @showtime_id, @promotion_id, @cust_id,
@ticket_count, @sale_amount;

WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT 'Showtime ID: ' + @showtime_id + ', Promotion ID: ' + @promotion_id
+
          ', Customer ID: ' + @cust_id + ', Ticket Count: ' +
CAST(@ticket_count AS VARCHAR(5)) +
          ', Sale Amount: ' + CAST(@sale_amount AS VARCHAR(10));

    FETCH NEXT FROM ticket_cursor INTO @showtime_id, @promotion_id, @cust_id,
@ticket_count, @sale_amount;
END

CLOSE ticket_cursor;
DEALLOCATE ticket_cursor;
```