**-- Supply Chain Analytics**

Create a Helper Table

```
-- Create fact_act_est table
	drop table if exists fact_act_est;

	create table fact_act_est
	(
	select
		s.date as date,
		s.fiscal_year as fiscal_year,
		s.product_code as product_code,
		s.customer_code as customer_code,
		s.sold_quantity as sold_quantity,
		f.forecast_quantity as forecast_quantity
	from
		fact_sales_monthly s
	left join fact_forecast_monthly f
	using (date, customer_code, product_code)
	)
	union
	(
	select
		f.date as date,
		f.fiscal_year as fiscal_year,
		f.product_code as product_code,
		f.customer_code as customer_code,
		s.sold_quantity as sold_quantity,
```

```
        f.forecast_quantity as forecast_quantity
    from
            fact_forecast_monthly  f
    left join fact_sales_monthly s
    using (date, customer_code, product_code)
    );

    update fact_act_est
    set sold_quantity = 0
    where sold_quantity is null;

    update fact_act_est
    set forecast_quantity = 0
    where forecast_quantity is null;
```

Database Triggers

-- create the trigger to automatically insert record in fact_act_est table whenever insertion happens in fact_sales_monthly

```
    CREATE DEFINER=CURRENT_USER TRIGGER `fact_sales_monthly_AFTER_INSERT`
AFTER INSERT ON `fact_sales_monthly` FOR EACH ROW
    BEGIN
    insert into fact_act_est
            (date, product_code, customer_code, sold_quantity)
```

```
        values (
        NEW.date,
        NEW.product_code,
        NEW.customer_code,
        NEW.sold_quantity
         )
        on duplicate key update
         sold_quantity = values(sold_quantity);
    END
```

-- create the trigger to automatically insert record in fact_act_est table whenever insertion happens in fact_forecast_monthly

```
    CREATE DEFINER=CURRENT_USER TRIGGER
`fact_forecast_monthly_AFTER_INSERT` AFTER INSERT ON `fact_forecast_monthly` FOR
EACH ROW
    BEGIN
    insert into fact_act_est
        (date, product_code, customer_code, forecast_quantity)
        values (
        NEW.date,
        NEW.product_code,
        NEW.customer_code,
        NEW.forecast_quantity
         )
        on duplicate key update
         forecast_quantity = values(forecast_quantity);
    END
```

-- To see all the Triggers

```
show triggers;
```

-- Insert the records in the fact_sales_monthly and fact_forecast_monthly tables and check whether records inserted in fact_act_est table

```
insert into fact_sales_monthly
 (date, product_code, customer_code, sold_quantity)
values
    ("2030-09-01", "HAHA", 99, 89);

insert into fact_forecast_monthly
 (date, product_code, customer_code, forecast_quantity)
values
    ("2030-09-01", "HAHA", 99, 43);

select * from fact_act_est where customer_code = 99;
```

Database Events

-- To show all the events

```
show events;
```

-- Show variable which have event in it

```
show variables like "%event%";


-- Creating the table "session_logs" in the random table and also insert the records in it
    CREATE TABLE random_tables.session_logs (`ts` DATETIME, `session_id` INT, `user_id` INT, `log` TEXT);
    INSERT INTO `random_tables`.`session_logs`
      (`ts`, `session_id`, `user_id`, `log`)
    VALUES
        ('2022-10-04 08:14:07', '898812', '523', 'CLICKED | Courses Buttom'),
    ('2022-10-14 08:18:35', '898812', '523', 'NAVIAGE BACK | Python course page , codebasics.io'),
    ('2022-10-16 12:07:00', '965345', '523', 'REVIEW GENERATED | Data analytics in power bi'),
    ('2022-10-22 14:09:22', '188567', '707', 'NEW LOGIN | New login, user name: tasty@jalebi.com'),
    ('2022-10-22 18:10:06', '188567', '707', 'COURSE PURCHASED | Data analytics in power bi, user name: tasty@jalebi.com');


-- Delete logs that are less than 5 days old
    delimiter |
    CREATE EVENT e_daily_log_purge
    ON SCHEDULE
    EVERY 5 SECOND
    COMMENT 'Purge logs that are more than 5 days old'
    DO
      BEGIN
    delete from random_tables.session_logs
    where DATE(ts) < DATE("2022-10-22") - interval 5 day;
      END |
```

```
delimiter ;

-- drop the event
drop event if exists e_daily_log_purge;
```

Temporary Tables & Forecast Accuracy Report

```
-- Forecast accuracy report using cte (It exists at the scope of statements)
with forecast_err_table as (
 select
    s.customer_code as customer_code,
    c.customer as customer_name,
    c.market as market,
    sum(s.sold_quantity) as total_sold_qty,
    sum(s.forecast_quantity) as total_forecast_qty,
    sum(s.forecast_quantity-s.sold_quantity) as net_error,
    round(sum(s.forecast_quantity-s.sold_quantity)*100/sum(s.forecast_quantity),1)
as net_error_pct,
    sum(abs(s.forecast_quantity-s.sold_quantity)) as abs_error,
    round(sum(abs(s.forecast_quantity-
sold_quantity))*100/sum(s.forecast_quantity),2) as abs_error_pct
 from fact_act_est s
 join dim_customer c
```

```
        on s.customer_code = c.customer_code
        where s.fiscal_year=2021
        group by customer_code
      )
      select
      *,
      if (abs_error_pct > 100, 0, 100.0 - abs_error_pct) as forecast_accuracy
      from forecast_err_table
    order by forecast_accuracy desc;

-- Write a stored proc for the same
      CREATE PROCEDURE `get_forecast_accuracy`(
      in_fiscal_year INT
      )
      BEGIN
            with forecast_err_table as (
               select
              s.customer_code as customer_code,
              c.customer as customer_name,
              c.market as market,
              sum(s.sold_quantity) as total_sold_qty,
              sum(s.forecast_quantity) as total_forecast_qty,
              sum(s.forecast_quantity-s.sold_quantity) as net_error,
              round(sum(s.forecast_quantity-
s.sold_quantity)*100/sum(s.forecast_quantity),1) as net_error_pct,
              sum(abs(s.forecast_quantity-s.sold_quantity)) as abs_error,
              round(sum(abs(s.forecast_quantity-
sold_quantity))*100/sum(s.forecast_quantity),2) as abs_error_pct
```

```
            from fact_act_est s
            join dim_customer c
        on s.customer_code = c.customer_code
        where s.fiscal_year=in_fiscal_year
        group by customer_code
      )
      select
      *,
      if (abs_error_pct > 100, 0, 100.0 - abs_error_pct) as forecast_accuracy
      from forecast_err_table
    order by forecast_accuracy desc;
  END

-- Forecast accuracy report using temporary table (It exists for the entire session)
    drop table if exists forecast_err_table;
    create temporary table forecast_err_table
     select
        s.customer_code as customer_code,
        c.customer as customer_name,
        c.market as market,
        sum(s.sold_quantity) as total_sold_qty,
        sum(s.forecast_quantity) as total_forecast_qty,
        sum(s.forecast_quantity-s.sold_quantity) as net_error,
        round(sum(s.forecast_quantity-s.sold_quantity)*100/sum(s.forecast_quantity),1)
as net_error_pct,
        sum(abs(s.forecast_quantity-s.sold_quantity)) as abs_error,
        round(sum(abs(s.forecast_quantity-
sold_quantity))*100/sum(s.forecast_quantity),2) as abs_error_pct
```

```
        from fact_act_est s
        join dim_customer c
        on s.customer_code = c.customer_code
        where s.fiscal_year=2021
        group by customer_code;

      select
      *,
      if (abs_error_pct > 100, 0, 100.0 - abs_error_pct) as forecast_accuracy
      from forecast_err_table
    order by forecast_accuracy desc;
```

User Accounts and Privileges

```
-- Show all grants available for a particular user(wanda)
        show grants for 'wanda';

-- Create a new user 'thor'
        create user 'thor'@'localhost' identified by 'thor';

-- Allow certain access to 'thor' user for the database 'gdb041'
        grant select on gdb041.dim_customer to 'thor'@'localhost';
        grant select on gdb041.dim_product to 'thor'@'localhost';
        grant execute on procedure gdb041.get_forecast_accuracy_report to
'thor'@'localhost';
```

```
-- See all the access for 'thor' user
	show grants for 'thor'@'localhost';
```

Database Indexes: Index Types (make sakila database as default one)

```
-- Query1
	select * from film where description like "%car%" or "%boat%";

-- Query2
	select * from sakila.film
	where match(description) against("car boat")
	limit 1000

-- Query3
	select * from sakila.film
	where match(description) against("car -boat" in boolean mode)
	limit 1000;
```