OBETA Script: ETL Implementation

1. Univariate Analysis	1.1 Overview of Each Variable and Define Key Variables with Statistical Description
2. Staging Phase	2.1 Create Tables in Database (MySQL)
3. Transformation Phase	3.1 Import Product Data and Pick Data from MySQL 3.2 Merge the Product and Pick Tables 3.3 Identify Dirty Data: *duplicated, missing, inconsistent values and outliers* 3.4 Clean the Data
4. Production DB	4.1 Create Dimensions Based on ER Model 4.2 Load Dimensions Tables Back to Production Database 4.3 Connect Production Database with Tableau
5. KPIs calculation	5.1 Update KPIs
6. Data Visualization	6.1 Create Dashboards Based on BI and KPIs 6.2 Create Tableau Story for Final Presentation

1.1 OVERVIEW OF EACH VARIABLE AND DEFINE KEY VARIABLES WITH STATISTICAL DESCRIPTION

Column Name	Variable Type	Key Variable	Statistical Description
product_id	Nominal	√	Count: 33493460 Unique Count: 97211 Top: 109910 Freq: 170062
description	Nominal		
product group	Nominal	√	Count: 33493460 Unique Count : 15 Top:. 31_ install. Freq: 8,760,367
product_id	Nominal		
warehouse section	Nominal	√	Count:33493460 Unique Count : 5 Top: SHL Freq: 14421610
origin (46:store, 48:customer)	Dichotomous	√	Count:33493460 Unique Count : 2 Top: "48" Freq: 217458068
order_no	Nominal	√	Count: 33493460 Unique Count : 9344863 Top: 201604055714 Freq: 371
position	Ordinal		
pick volume	Metric	√	Count: 33493460 Mean: 45.93 Median: 5 Mode: 1 SD: 127.42 Min: -200 Max: 1160
qty_unit	Nominal		
date	Ordinal		

2.1 CREATE TABLES IN DATABASE (MySQL)

Purpose	Query
Create product table in "OBETA" schema	Creation of Product data Table CREATE TABLE product_data (productid INT, product_name VARCHAR (50) NULL, product_group VARCHAR (100) NULL);
Create pick table in "OBETA" schema	Create schemas

	CREATE SCHEMA `staging_data` ; CREATE TABLE product_pick (productid VARCHAR (100) NULL, warehouse_area VARCHAR (100) NULL, order_origin VARCHAR (100) NULL, order_number VARCHAR (100) NULL, position_in_order VARCHAR (100) NULL, volume_of_order INT NULL, quantity_unit VARCHAR (50) NULL, date_of_order DATETIME);
Import product.csv into MySQL	LOAD DATA local INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\product_data.csv" INTO TABLE product_data CHARACTER SET latin1 FIELDS TERMINATED BY ',' ENCLOSED BY " LINES TERMINATED BY '\n' IGNORE 1 ROWS; set global local_infile=1;
Import pick.csv into MySQL	LOAD DATA local INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\pick_data.csv" INTO TABLE pick_data FIELDS TERMINATED BY ',' ENCLOSED BY "" LINES TERMINATED BY '\n'; set global local_infile=1;

3.1 IMPORT PRODUC DATA AND PICK DATA FROM MySQL

Purpose	Platform/Libraries	Query
Connect Rstudio to MySQL (Database) and Copy Datasets	Rstudio/DBI, RMySQL, dplyr	# connect to staging data Schema pro_con<- dbConnect(RMySQL::MySQL(),

		dbname="staging_data",
		host='localhost',
		port=3306,
		user='root',
		password='mysql')
		
		dbListTables(pro_con, 'product_data')
		
		#call product data from mysql
		prod_d <- tbl(pro_con, "product_data")
		
		#copy product data to R global env't
		prod_df <- collect(prod_d)
		
		#call pick data from mysql
		pick_d <- tbl(pro_con, "pick_data")
		
		#copy pick data to R global env't
		pick_df <- collect(pick_d)

3.2 MERGE THE PRODUCT AND PICK TABLES WITH RSTUDIO

Purpose	Query
Join the Product and Pick datasets	joined_table <- inner_join (pick_df, prod_df, by = "product_id")

3.3 IDENTIFY DIRTY DATA WITH RSTUDIO

Purpose	Query

Find duplicated rows	#Create a DF and flag for duplicated rows
	joined_df_dupli <- joined_df %>%
	group_by_all() %>%

	mutate(is_duplicate = n() > 1)
	
	table(joined_df_dupli$is_duplicate)
	
	# filter duplicate rows
	only_dupli_rows <- joined_df %>%
	group_by_all() %>%
	filter(n() > 1)

Find missing values	missing_values <- joined_df %>%
	summarise_all(~ sum(is.na(.)))
	
	#Find missing strings in all columns
	missing_strings <- joined_df %>%
	summarise_all(~ sum(. == ""))

Find duplicated order IDs (more than one order year)	#Create a order year column
	joined_table$order_year <- year(joined_table$date_of_order)
	
	#filter out order IDs which duplicate themselves in more than one order year
	duplicated_orderid <- joined_table %>%
	group_by(order_number) %>%
	filter(n_distinct(order_year) > 1) %>%
	ungroup()
	table()

Find rows with zero order volumes	#Filter rows with zero order volumes
	zero_values<- filter(joined-table, volume_of_order == 0)

Find rows with negative order volumes	#Filter rows with negative order volumes
	negative_values<- filter(joined-table, volume_of_order < 0)
	
	table(zero_values)

Detect outliers	# Create outlier flag function
	outlier.create.flag<-function(x)
	{hist(x)

	summary(x)
	
	# using here the z values
	flag<- ifelse(test = scale(x) > 3 | scale(x)< -3, yes = 1, no = 0)
	# number of outlier
	table(flag)
	
	# return the vector representing the flag variable
	flag}
	
	# Create outlier column for joined data and add to the dataframe
	join_non_zero$volume.outlier.flag<-
	outlier.create.flag(join_non_zero$volume_of_order)
	
	table(join_non_zero$volume.outlier.flag)
	
	#create only outlier table on R
	
	only_outlier_table <- filter(joined_df, volume.outlier.flag ==1)

3.4 CLEAN THE DATA WITH RSTUDIO

Purpose	Query
Remove duplicated rows	join_df_unique <- joined_df %>% distinct()
Remove rows with zero value	join_non_zero <- join_df_unique %>% filter(volume_of_order != "0")
Create unique order ID for duplicated values	#Merge year of order and order ID join_non_zero$orderid_unique<- do.call(paste, c(join_non_zero [column_merge], sep=""))
Create a filtered table to exclude outliers	join_clean_df <- filter(join_non_zero, volume.outlier.flag ==0)

4.1 CREATE DIMENSIONS BASED ON ER MODEL

Purpose	Platform/Libraries	Query

Create Warehouse Section Dimension Table	Rstudio/dplyr	#create warehouse name and column
		join_facts <- join_facts %>%
		mutate(section_name = case_when(
		grepl("AKL", warehouse_area, ignore.case = TRUE) ~ "Automated Small-Parts Warehouse",
		grepl("HRL", warehouse_area, ignore.case = TRUE) ~ "High Bay Storage for Pallets",
		grepl("SHL", warehouse_area, ignore.case = TRUE) ~ "Shuttle house",
		grepl("Kabellager", warehouse_area, ignore.case = TRUE) ~ "Cable Storage",
		grepl("Manuell", warehouse_area, ignore.case = TRUE) ~ "Manual Warehouse",
		TRUE ~ "Other"))

Create Product Dimension Table	Rstudio	product_dim <- join_facts[,c("product_id", "description", "product_group",
		"quantity_unit")]
		#remove duplicated rows
		product_dim <- product_dim %>% distinct()

Create Order Dimension Table	Rstudio	#Create Order dimension table
		
		order_dim <- join_facts[,c("orderid_unique", "order_number", "order_origin",
		
		"position_in_order", "date_of_order")]
		
		#remove duplicated rows
		order_dim <- order_dim %>% distinct()

Create Time Dimension Table	Rstudio/lubridate	#Create Time Dimension
		time_dim <- join_facts[,c("date_of_order", "order_year")]
		
		# Extract day
		time_dim$day <- day(time_dim$date_of_order)

		#extract_month time_dim$month <- month(time_dim$date_of_order)

Create Facts Table	Rstudio	#Create Facts Table
		facts_table <- join_facts[,c("warehouse_area","orderid_unique", "product_id",
		"date_of_order", "position_in_order",
		"volume_of_order", "pick_efficiency")]

4.2 LOAD DIMENSION TABLES BACK TO PRODUCTION DATABASE

Purpose	Platform/Libraries	Query

Connect Rstudio to Production Schema on MySQL	Rstudio/DBI, RMySQL, dplyr	# connect to production data Schema
		prod_con<- dbConnect(RMySQL::MySQL(),
		dbname="production_data", host='localhost',
		port=3306,
		user='root',
		password='mysql')

Copy Warehouse Section Dimension Table into MySQL's Schema_Production Data	Rstudio/DBI, RMySQL, dplyr	dbWriteTable(prod_con,"warehouse_section_dim",warehouse_section_dim,row.names=FALSE)
Copy Product Dimension Table into MySQL's Schema_Production Data	Rstudio/DBI, RMySQL, dplyr	dbWriteTable(prod_con,"product_dim",product_dim,row.names=FALSE)
Copy Order Dimension Table into MySQL's Schema_Production Data	Rstudio/DBI, RMySQL, dplyr	dbWriteTable(prod_con,"order_dim",order_dim,row.names=FALSE)
Copy Time Dimension Table into MySQL's Schema_Production Data	Rstudio/DBI, RMySQL, dplyr	dbWriteTable(prod_con,"time_dim",time_dim,row.names=FALSE)

5.1 UPDATE KPIs

Purpose	Query
Pick Efficiency per Product	facts_table <- facts_table %>% arrange(orderid_unique, date_of_order) %>% group_by(orderid_unique) %>% mutate(effi2 = as.numeric(difftime(date_of_order, lag(date_of_order, default = first(date_of_order)), units = "mins")))
Pick Efficiency per Order	Create a calculated field for sum of Group Efficiency Sum { FIXED [orderid_unique] : SUM([Pick Efficiency Kpi]) } create a calculated field for Average pick efficiency WINDOW_AVG([Group Efficiency Sum])